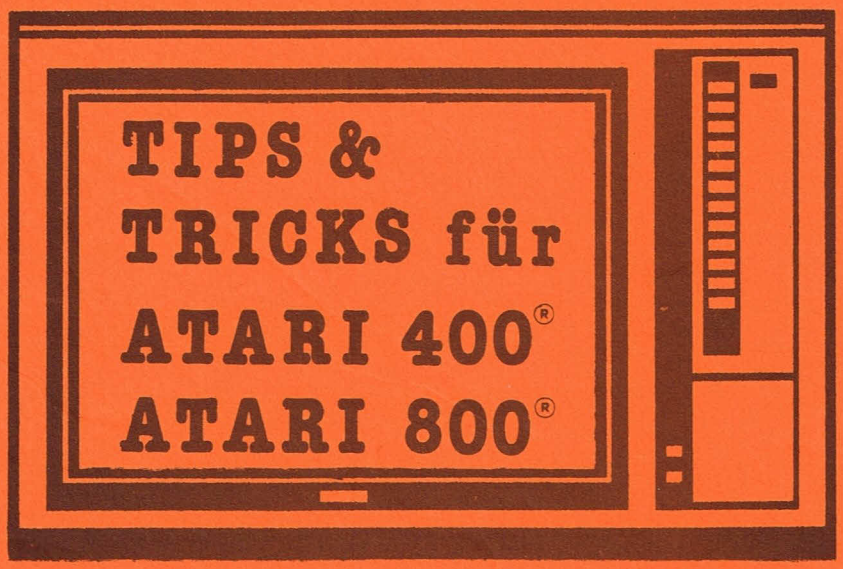
Die Fundgrube



ALLEN ATARI 400/800 - BESITZERN,

DIE MIT IHREN COMPUTERN MEHR TUN,

ALS NUR ZU SPIELEN...

I M P R E S S U M

H. Zoschke: DIE FUNDGRUBE. TIPS UND TRICKS FÜR ATARI 400 UND ATARI 800

(C) 1982 by

Ing.-Büro für technische Dokumentation
Dipl.-Ing. Harald Zoschke
Kleinhartpenning 7a
D-8150 Holzkirchen 2

Sämtliche Rechte (auch die des auszugsweisen Nachdrucks, der Fotokopie, der Übersetzung und der Speicherung auf magnetischen und sonstigen Trägern) vorbehalten. Die in diesem Buch wiedergegebenen Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt; sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden. Für Folgen, die auf fehlerhafte Angaben zurückzuführen sind, kann weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung übernommen werden. Dies gilt ebenfalls für sämtliche Nachteile wie Garantieverlust, Beschädigungen etc. die durch Öffnen von Originalgeräten und deren unautorisierter Manipulation entstehen.

Für eine Mitteilung eventueller Fehler sowie Anregungen ist der Autor jederzeit dankbar (Anschrift s.o.)

ATARI, ATARI 400, -410, -800, -810, -830 und -850 sind eingetragene Warenzeichen von ATARI Inc., Sunnyvale, CA 94086 (BRD: Bebelalle 10, D-2000 Hamburg 60), einer Gesellschaft der Warner Communications.

1. Auflage 1982

I N H A L T

VORWORT

TEIL 1: DATENSEPARATOR FÜR ATARI 810 3

Nachträglicher Einbau eines Datenseparators 4

DISK TEIL 2: ROM-ÄNDERUNGEN IN DER ATARI 810 6

Der Einbau neuer ROMs und EPROMs 6

DISK TEIL 3: DAS INNENLEBEN DER DISKETTENSTATION ATARI 810

ATARI 810 - Ein Diskettenlaufwerk mit eigenem Computer 8
Blockschaltung des 810-Systems 9
Memory Map für ATARI 810 10
Schaltplanauszug 11
Hardwarezusatz zum Beschreiben der Disketten-Rückseiten 12
Netzteilprobleme bei ATARI 810 13
Tips zum Thema Stromversorgung 14

DISK TEIL 4: WARTUNG DER DISKETTENSTATION ATARI 810 15

Schmierung 15
Kopf-Reinigung 16
Federdruck-Kontrolle 16
Drehzahleinstellung per Software 17
Kopf-Entmagnetisierung 18
Vorbeugende Wartung 19

SCHNITTSTELLENPROGRAMMIERUNG TEIL 1: DIE STEUERKNÜPPEL-PORTS 20

SCHNITTSTELLENPROGRAMMIERUNG TEIL 2: DIE DREHREGLER-ANSCHLÜSSE ... 24

WISSENSWERTES ÜBER EPROMS

Grundsätzliches 26
Wie wird ein EPROM programmiert? 28
Wie löscht man ein EPROM? 29
Tips zum Umgang mit EPROMs 30
Übersicht: Anschlußbelegung von ROMs und EPROMs 31
Umbau eines ATARI-ROM-Moduls zur Aufnahme von EPROMs 32

ZAHLENWANDLUNG TEIL 1: DIE TABELLE 36

ZAHLENWANDLUNG TEIL 2: PROGRAMMBEISPIELE IN BASIC 37

'ZU FUß' DISASSEMBLIEREN MIT DER OPERATIONSCODE-TABELLE! 38
MONKEY WRENCH (tm) BEFEHLSÜBERSICHT 40

DOS-TIPS 41

STECKVERBINDER FÜR ATARI: TYPENBEZEICHNUNGEN 43

I N H A L T

STECKVERBINDER FÜR ATARI: ANSCHLUSSBELEGUNGEN UND -KABEL

Anschluß von Drucker, Modem etc. 44
Anschluß des ATARI 800 an Video-Monitor und Stereo-Anlage 45

SOFTWARE-SCHUTZ

List- und Kopierschutz-Tricks 46
Schlüsselwort-Routinen 47

DIE MEMORY MAP ODER 'WAS LIEGT WO IM ATARI?'

Graphische Darstellung mit Speicherkarten 49
Hex-Dump in BASIC 52

DIE MEMORY MAP ODER WAS LIEGT WO IM ATARI TEIL 2:

Alle wichtigen Peek- und Poke-Adressen 53

BEISPIEL FÜR DIE PROGRAMMIERUNG DER DREHREGLERANSCHLÜSSE:
HELLIGKEITSSTEUERUNG

Problemlösung in BASIC und Maschinensprache 74
Interruptgesteuerte Helligkeitssteuerung - zugleich Beispiel
für die Benutzung von Timern und Interrupts 76

ROM-MODULE SELBERMACHEN!

Grundlagen: Vektoren und alles, was dazugehört 78
Ein konkretes Beispiel 79

DISK TEIL 5: DER AUTORUN.SYS-GENERATOR 82

ERROR 144 ETC.: FEHLER SCHNELL GEFUNDEN UND BESEITIGT! 83

Teil 1: Allgemeine Tips zum Umgang mit Fehlern 84
Teil 2: Fehlerliste und Erklärungen 85
Teil 3: Hilfsprogramm zur Ausgabe von Klartext-Fehlermeldungen . 103
Fehlerbehandlung in Maschinensprache 103

BASIC-PROGRAMME AUFPOLIEREN: EINBAUFERTIGE SOUND-ROUTINEN 105

FACHBÜCHER UND ZEITSCHRIFTEN FÜR ATARI-COMPUTER 106

SCHNITTSTELLENPROGRAMMIERUNG TEIL 3

Die serielle System-Schnittstelle 107
Die TRIG-Eingänge 108

SAMMELSURIUM - BUNT GEMIXTES FÜR IHREN ATARI 109

V O R W O R T

Dieses Buch, lieber ATARI-Besitzer, hat einen etwas unkonventionellen Charakter. Es ist weder im Fotosatz gedruckt, noch ist es säuberlich in Kapitel unterteilt. Beides hat seine Gründe. Um den Preis für Sie niedrig zu halten, erfolgte der Ausdruck mit einem Typenraddrucker. Und um Ihnen die hier präsentierten Tips und Tricks unabhängig von Interessen Dritter mitteilen zu können, geschah die gesamte Herstellung unter 'eigener Regie'.

Zum anderen handelt es sich hier nicht um ein Lehrbuch, sondern eben um eine FUNDGRUBE. Beim Stöbern werden Sie viele interessante Dinge finden, die bislang nur wenigen Insidern bekannt waren. Vieles wurde erst durch Nachforschungen vor Ort in USA zutage gefördert. Einiges wird bei Ihnen wahrscheinlich schnell zum häufig benutzten Nachschlagewerk werden: zum Beispiel die ausführlichen Erklärungen der - leider nicht immer vermeidbaren - Fehlermeldungen. Oder die Memory Map mit (hoffentlich) allen wichtigen Adressen im ATARI-System. Es hindert Sie übrigens niemand daran, eigene Ergänzungen vorzunehmen!

Abschließend noch ein paar Hinweise:

1) Einige Vorschläge in diesem Buch sind mit Eingriffen, z.B. in die Disketten-Station, verbunden. Neben dem Verlust der Garantie (sofern noch welche besteht) können unsachgemäß durchgeführte Arbeiten natürlich mehr Schaden anrichten als Verbesserungen bringen. Es sei daher nochmals ausdrücklich auf den Haftungsausschluß im Impressum hingewiesen. Aber keine Angst: Alles wurde in unserem Labor erprobt!

2) Entgegen ihrem Mitteilungszweck sind einige Informationen in diesem Buch dazu geeignet, unerlaubte Programm-Kopien anzufertigen. Bitte bedenken Sie, welche unvorstellbaren Kettenreaktionen damit ausgelöst werden können. Und es dürfte sicherlich im Interesse keines ATARI-Anwenders liegen, daß sich Produzenten hervorragender Software - egal ob hier oder in den USA - aus diesem Grund aus dem Geschäft zurückziehen.

3) Alle DRUCKFEHLER sind beabsichtigt. Denn die FUNDGRUBE soll ja für jeden etwas bieten - auch für die Leute, die nur nach Fehlern suchen.

DIE MÖGLICHKEITEN DES ATARI-SYSTEMS SIND NOCH LÄNGST NICHT AUSGESCHÖPFT. DIE FUNDGRUBE SOLL IHNEN DABEI HELFEN, NOCH MEHR AUS IHREM COMPUTER HERAUSZUHOLEN.

Holzkirchen
Herbst 1982

Harald Zoschke

DISK TEIL1: DATENSEPARATOR FÜR ATARI 810

ATARI-Freunde, die schon länger in Besitz einer Diskettenstation waren und sich im Winter '81 oder später ein zweites Laufwerk anschafften, waren überrascht: nicht nur, daß sich Disketten schneller formatieren und laden ließen. Auch die Rate der Lesefehler, gerade bei fremden Disketten, ging mit dem neuen Gerät praktisch auf Null zurück. Besonders Neugierige schraubten ihre Diskettenstationen unter dem Risiko des Garantie-Verlustes auf, um das Innenleben zu vergleichen. Auf den ersten Blick nichts Neues. Das Geheimnis lüftet sich erst mit dem Öffnen der Blech-Abschirmung auf der Seitenplatine. Hier hat sich einiges getan:

1. Ein neues Betriebssystem-ROM für den 6507, inzwischen bekannt als 'Fast Formatting Chip' (siehe 'DISK TEIL2').

2. Eine kleine Platine, die in Sandwich-Bauweise auf dem Sockel des 40-poligen Steuerungs-ICs 1771 steckt.

Obwohl nirgendwo erwähnt oder vermerkt, ließ sich schnell feststellen, daß es sich bei dieser kleinen Zusatzschaltung um einen DATENSEPARATOR handelte. Alle Geräte, die diese beiden Änderungen enthielten, waren ohne weitere Erläuterungen mit zwei Aufklebern in ihrem Inneren gekennzeichnet: 'DS' (für Datenseparator) und 'C' (für Revision C). Alle ab 1982 produzierten Geräte sollten eigentlich so ausgestattet sein. Durch Zwischenlagerung bei der deutschen ATARI-Tochter und/oder beim Fachhändler bedingt sind jedoch auch noch 'alte' Geräte in Umlauf.

WELCHE VERBESSERUNGEN BRINGT DER DATENSEPARATOR?

Von allen Verbesserungen, die der Disk 810 im Laufe ihres Daseins widerfahren sind, dürfte der Einbau eines Datenseparators die wirksamste sein. Selbst auf den bisher besonders kritischen inneren Diskettenspuren gibt es kaum noch Lese-Fehler. Auch lassen sich auf anderen Laufwerken bzw. mit zu hoher oder zu niedriger Drehzahl bespielte Disketten endlich problemlos laden. Ebenso ist das Schreiben sicherer geworden, und das Formatieren einer Diskette klappt nun meist beim ersten Versuch. Darüberhinaus erhöht sich die Zahl der Disketten, die sich auch auf der Rückseite bespielen läßt (siehe 'DISK TEIL 3').

WIE STELLE ICH FEST, OB ICH BEREITS EINE DISKETTENSTATION MIT DATENSEPARATOR BESITZE?

Ganz einfach: Sie nehmen einen Kreuzschlitzschraubenzieher und öffnen das Gehäuse Ihrer ATARI 810. Sie finden keine Schrauben? Kein Wunder, sie sind nämlich zur Tarnung mit Kunststoffaufklebern verdeckt, die Sie zunächst mit einem Messer vorsichtig entfernen müssen (siehe Abb. 1).

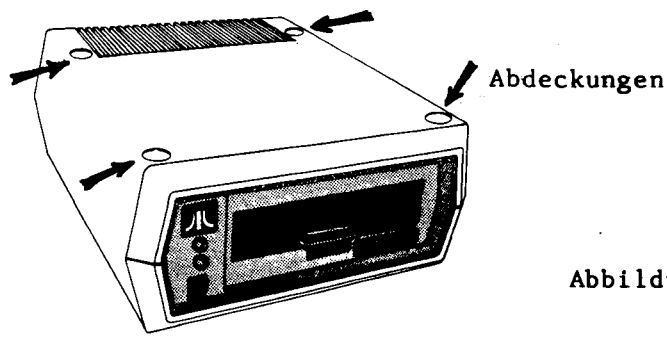


Abbildung 1

Finden Sie nach Herausdrehen der Schrauben und Abnehmen des Deckels irgendwo im Inneren Ihres Laufwerkes die besagten Aufkeber 'DS' und 'C', ist der Fall klar. Sie können das Gerät wieder zuschrauben. Wenn nicht, hilft nur weitersuchen! Machen Sie dazu alle Schrauben ausfindig, die Sie zum Herausnehmen der Seitenplatine lösen müssen. Nochmals sei daran erinnert, daß durch derartige Eingriffe die Garantie erlischt, sofern sie nicht sowieso schon abgelaufen ist.

Bevor Sie die Seitenplatine herausnehmen, ziehen Sie die beiden Stecker ab und merken sich deren Position. Ziehen Sie das Board von seinem Stecksockel auf der Grundplatine und entfernen Sie die Metallabschirmung, indem Sie die zwei Schrauben lösen und vorsichtig alle Blechlaschen hochbiegen. Steckt das 40-polige IC '1771' direkt in seinem Sockel auf der Platine, sind Sie tatsächlich noch kein glücklicher Besitzer eines Datenseparators. Also alles wieder zusammenbauen und weitermachen wie bisher, bis Sie einen haben.

WOHER BEKOMME ICH EINEN DATENSEPARATOR?

In den USA ist bereits eine entsprechende ATARI-Platine als Nachrüstsatz zu haben. Aber auch dort gab es Beschaffungsschwierigkeiten, die zur Entdeckung einer zudem noch preiswerteren Alternative führten:

Bereits seit mehreren Jahren gibt es von PERCOM einen Datenseparator für die TRS-80-Disk, der sich auch im ATARI-810-Laufwerk einsetzen läßt. PERCOM bietet inzwischen sogar eine entsprechende Einbauanleitung an. In den US-Computershops wird dieser Zusatz z.Zt. für knapp 30 Dollar verkauft. Aber auch ohne die Anleitung ist der Einbau denkbar einfach, sodaß man bei den heimischen TRS-80-Händlern nachfragen sollte, bevor man sich zu einer kosten- und zeitaufwendigen Bestellung in USA entschließt.

EINBAU DES PERCOM-DATENSEPARATORS

1. Bauen Sie die Seitenplatine auf die bereits beschriebene Weise aus und entfernen Sie die Blechabschirmumg.

2. Hebeln Sie das 40-polige IC '1771' mit einem Schraubenzieher von beiden Seiten vorsichtig aus seinem Sockel.

ACHTUNG: Da es sich um einen elektrostatisch empfindlichen MOS-Baustein handelt, sollte man die IC-Anschlüsse möglichst nicht mit den Fingern berühren.

3. Stecken Sie das IC in die 40-polige Fassung auf der Datenseparator-Platine. Achten Sie dabei auf die Kerbe, damit dies nicht falsch herum geschieht.

4. Aus mechanischen Gründen muß der Quarz auf der Seitenplatine <u>liegend</u> montiert werden. Löten Sie ihn dazu aus, löten Sie zur Verlängerung der Anschlüsse etwas Draht an und bauen Sie ihn wie in Abbildung (3) gezeigt wieder ein.

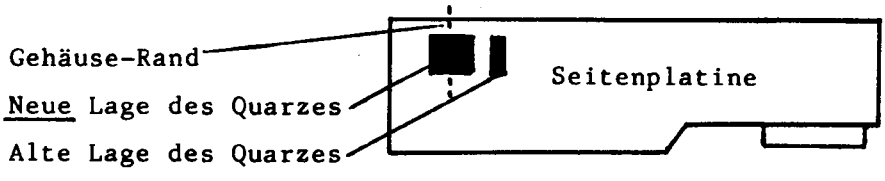


Abbildung (3)

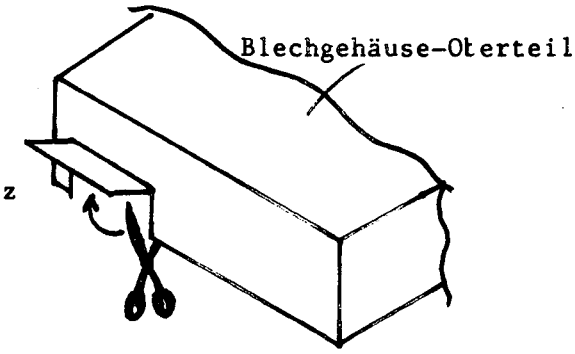


Abbildung ④

5. Wegen der Umlegung des Quarzes ist ein kleiner Ausschnitt im Abschirmungsgehäuse erforderlich. Nehmen Sie mit der oberen Abdeckung des Blechgehäuses Maß, machen Sie mit einer Schere zwei senkrechte ca. 8 mm tiefe Einschnitte und biegen Sie das Blech wie in Abbildung ④ gezeigt hoch.

6. Stecken Sie das Datenseparator-Board richtig herum in die Seitenplatine. – ACHTUNG: Um ungewünschten Kontakt zwischen den beiden Platinen zu vermeiden, sollte man deren Abstand mit Hilfe eines 40-poligen Zwischensockels vergrößern. Verwenden Sie dazu möglichst eine Fassung mit gedrehten Kontakten (sog. Kelchfederkontakte).

7. Setzen Sie alles wieder so zusammen, daß möglichst keine Teile übrigbleiben!!

Falls Sie bei TRS-80-Händlern keinen Erfolg haben sollten, hier die Anschrift von PERCOM:

PERCOM DATA COMPANY INC.
11220 Pagemill Rd.
Dallas/Texas 75243
Tel.: (214)340-7081

Sollten Sie keine Bezugsquelle für Datenseparatoren ausfindig machen, hilft Ihnen der Autor gerne weiter.

Übrigens: PERCOM bietet seit Kurzem ein eigenes Disketten-System an, und zwar unter der Bezeichnung RFD.

Ein Kurzsteckbrief des RFD-Disketten-Systems:

- Ein Hauptlaufwerk mit der Steuerlogik, bis zu drei (billigere) Zusatz-Laufwerke
- Mit ATATRI 810 kombinierbar
- Kompatibel zu DOS2.OS
- Einfache und doppelte Aufzeichnungsdichte (umschaltbar)
- 40 Spuren, 184 kByte formatierte Speicherkapazität
- 24 kByte RAM erforderlich
- Preis (10/82): Hauptlaufwerk ca. 800 Dollar
  Zusatzlaufwerke ca. 460 Dollar

Nicht uninteressant!

DISK TEIL2: ROM-ÄNDERUNGEN IN DER DISKETTENSTATION ATARI 810

Als zweite wichtige Verbesserung der Disk 810 wurde ab Herbst '81 die Installation des 'ROM C', inzwischen auch als 'Fast Formatting Chip' bekannt, vorgenommen. Es scheint so, als ob alle neu ausgelieferten Laufwerke grundsätzlich beide Neuerungen enthielten.

WELCHE VERBESSERUNGEN BRINGEN DIE NEUEN ROMS?

Der Name sagt es bereits: die Disketten werden mit diesem ROM derart formatiert, daß sie schneller beschrieben und gelesen werden können. Entscheidend scheint dabei allein der Formatierungsprozeß zu sein, denn Disketten, die auf einer 'schnellen' Maschine formatiert wurden, lassen sich auch auf 'alten' Laufwerken mit gesteigerter Geschwindigkeit lesen und beschreiben. Grobe Messungen mit der Stopuhr ergaben ein ca. zwei- bis dreimal schnelleres Laden von Disketten-Daten!

WOHER BEKOMME ICH DIE NEUEN ROMS?

Wie Sie sehen, ist gleich von mehreren neuen ROMs die Rede. Da ATARI zunächst alle neuen 'ROM C'-Bausteine intern für die laufende 810-Produktion benötigte, waren sie als Ersatzteile lange nicht vefügbar. Da man nun aber gesehen hatte, das die Disk auch schneller kann, wenn sie entsprechend programmiert wird, entwickelten findige Tüftler in amerikanischen ATARI-Clubs ihre eigenen 'Fast Formatting Chips', die eine mindestens genauso große Geschwindigkeitssteigerung bewirken. Wer weder bei einem der US-Clubs noch bei seinem heimischen ATARI-Händler Erfolg hat, bekommt für ca. 45 Dollar ein schnelles ROM unter der Bezeichnung 'FAST-CHIP' bei:

BINARY Corp., 3237 Woodward Ave., Berkley, MI 48072, Tel. (313)548-0533.

WIE BAUE ICH EIN NEUES ROM EIN?

Wer ein original ATARI-ROM (2316) bekommt, braucht lediglich das auf der Seitenplatine mit A102 gekennzeichnete 24-polige IC gegen den neuen Baustein auszutauschen. Achten Sie auf die Kerbe! Öffnen des Gehäuses und Freilegen der Seitenplatine siehe 'DISK TEIL1'.

Viele Anbieter liefern ihr neues Disk-Betriebssystem jedoch in EPROMs vom Typ 2516/2716. Und wer ein Erstgerät mit langsamem und ein Zweitlaufwerk mit schnellem ROM besitzt, darf für seinen privaten Bedarf natürlich ohne weiteres eine EPROM-Kopie vom neuen Baustein erstellen, um die 'alte' Diskettenstation damit aufzurüsten. Wer über ein EPROM-Programmiergerät verfügt, kann das 2 kByte große Betriebssystem der Disketten-Elektronik auch disassemblieren und ändern, um als Programmschutz-Maßnahme bereits beim Formatieren Sektoren auszulassen ('Bad Sectoring'), etc.!!! Interessant ist also auch die Frage:

WIE BAUE ICH EIN EPROM EIN?

Aufgrund der geringfügig differierenden Anschlußbelegung des EPROM 2516(2716) gegenüber dem Masken-ROM 2316 wären eigentlich einige Modifikationen auf der Seitenplatine erforderlich. Ein kleiner Trick macht es jedoch möglich, daß auf der Platine nur ein einziger Draht angelötet bzw. angesteckt werden muß (siehe nächste Seite).

Gehen Sie zum Einsatz eines Eproms vom Typ 2516/2716 in Ihrer ATARI 810 wie folgt vor:

1. Programmieren Sie Ihr EPROM mit dem gewünschten Inhalt.

2. Zwicken Sie mit Hilfe eines kleinen Seitenschneiders die Anschlußbeinchen von Pin 18,20 und 21 ab, sodaß sie beim Einstecken des ICs in den Sockel dort keinen Kontakt mehr bekommen (siehe Abbildung 5).

3. Löten Sie ein Stück dünnen Draht von Pin 18 zu Pin 12, sowie von Pin 21 zu Pin 24.

4. Löten Sie einen ca. 70 mm langen dünnen isolierten Draht an Pin 20 und isolieren Sie auch das andere Ende ab (ca. 3 mm.)

WICHTIG: Verwenden Sie für ALLE Lötarbeiten einen Niedervolt-Lötkolben! Andernfalls können die empfindlichen MOS-IC beschädigt werden.

5. Entfernen Sie das original ATARI-ROM (Bezeichnung auf der Platine: A102) aus seinem Sockel und setzen Sie das modifizierte EPROM ein.

6. Verbinden Sie den an Pin 20 angeschlossenen Draht mit Pin 2 von Z103 (siehe Abbildung 6). Dies kann entweder durch Anlöten geschehen, oder durch Einstecken in den Sockelkontakt (Z103 dazu vorher herausnehmen). Sicherer ist natürlich die Lötverbindung.

7. Das wärs. Bevor Sie mechanisch alles wieder montieren, sollten Sie zuerst eine Probe machen, indem Sie die Platine wieder in ihren Sockel stecken, die beiden Steckverbinder aufstecken und das Laufwerk ausprobieren.

Soll das modifizierte EPROM einmal neu programmiert werden, bedient man am Besten eines Zwischensockels, mit dessen Hilfe die 'normalen' Anschlußverhältnisse vorrübergehend wieder hergestellt werden.

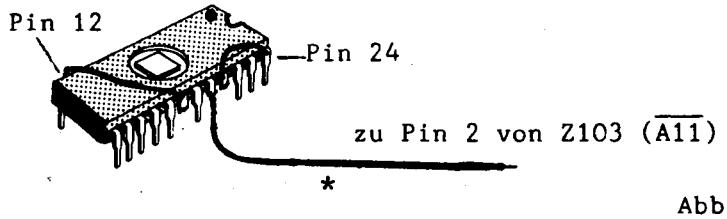


Abbildung 5
Modifiziertes EPROM als ROM-Ersatz in ATARI 810

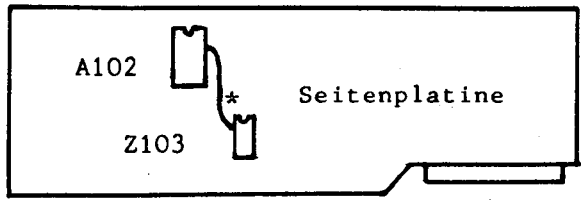


Abbildung 6
Installation des modifizierten EPROMs auf der Seitenplatine

DISK TEIL3: DAS INNENLEBEN DER DISKETTENSTATION ATARI 810

Diesem Abschnitt muß vorrausgeschickt werden, daß es extrem schwierig ist, technische Informationen über das Innenleben der ATARI 810 zu bekommen. Die hier mitgeteilten Erkenntnisse beruhen daher hauptsächlich auf dem Studium der Leiterplatte, des ROM-Betriebssystems und der Datenblätter zu den verwendeten LSI-Chips. Umsomehr gilt hier daher der Hinweis zu Beginn dieses Buches, daß für die Richtigkeit der Information und für eventuelle Folgeschäden durch Falschinformation keine Verantwortung übernommen wird.

Desweiteren beschränken wir uns bei der Hardware-Darstellung auf jene Details, die zur Modifikation des Betriebssystems nötig sind. Solange ATARI kein Hardware-Manual für die Disk 810 veröffentlicht, möge Ihnen die folgende Information von Nutzen sein.

ATARI 810 - Ein Diskettenlaufwerk mit eigenem Computer

Während die meisten anderen Personal-Computer für die Disketten-Handhabung eine ganze Menge Rechenzeit abzweigen müssen, besitzt die ATARI 810 ihren eigenen kompletten Computer, der den Hauptrechner kräftig entlastet.

Natürlich handelt es sich hier um eine Minimalkonfiguration, also um einen 'abgemagerten' Computer, der ganz speziell auf seine Aufgaben zugeschnitten ist: die Steuerung aller Laufwerk-Funktionen sowie die Datenübertragung vom und zum Hauptrechner. Das Herz des Systems bildet ein 'kleiner Bruder' des im ATARI 400/800 verwendeten 6502-Mikroprozessors: der 6507. In einem kleineren Gehäuse untergebracht, hat er nur 13 Adreßleitungen, mit denen er lediglich 8 kByte adressieren kann (6502: 16 Leitungen/64 kByte). Aber das ist sowieso schon mehr Speicherplatz, als das System benötigt. Außerdem fehlen ihm die beim 6502 vorhandenen Interrupt-Leitungen. Aber auch auf die kann in diesem Fall verzichtet werden. Im System finden sich einige LSI-Bausteine, die dem 6507 viel Arbeit abnehmen, an erster Stelle der Floppy-Disk-Controller 1771 von Western Digital. Fast schon selbst ein Mikroprozessor, versteht er eine Reihe von Befehlen, die ihn zur Ausführung ganzer Ketten von Operationen befähigen. So übernimmt er z.B. die Formatierung, das Lesen und Schreiben der Sektoren etc.

Als Schreib-Lese-Speicher benötigt das System lediglich 256 Bytes RAM. Davon enthält der RAM-Baustein 6810 (Motorola u.a.) 128 Bytes und der Multifunktions-Chip 6532 (Synertek u. a.) weitere 128 Bytes. Darüberhinaus stellt der 6532 zwei programmierbare Timer und - ähnlich wie in der PIA 6520 - 16 Schnittstellen-Leitungen zur Verfügung, die wahlweise als Ein- oder Ausgänge programmmiert werden können. Das Betriebssystem steht, wie wir ja schon wissen, in einem 2716-kompatiblen ROM.

Ein Blick ins Gehäuse-Innere zeigt, daß die Elektronik der ATARI 810 auf zwei Platinen verteilt ist. Auf der Bodenplatine befinden sich das Netzteil (+/-5V,+12V), die Tacho-Regelung für den Spindelmotor, die Ansteuerlogik für Löschkopf und Schreib-Lese-Kopf, die Laufwerkauswahl-Schaltung mit den Kodierschaltern und alle Anschlußbuchsen. - Auf der Seitenplatine, befindet sich größtenteils mit einem Blechgehäuse abgeschirmt, der Computerteil der Steuerung, bestehend aus CPU, Floppy-Controller, RAM, ROM, E/A-Chip, Taktgenerator und etwas CMOS-Logik zur Adreßkodierung. Vom E/A-Chip führen Leitungen zur Vierphasen-Steuerung des Schrittmotors für die Kopf-Positionierung, zum Floppy-Controller-Ic und zu den parallelgeschalteten Anschlußbuchsen für die serielle System-Schnittstelle.

Grob vereinfacht ergibt sich für die ATARI 810 das folgende Blockschaltbild:

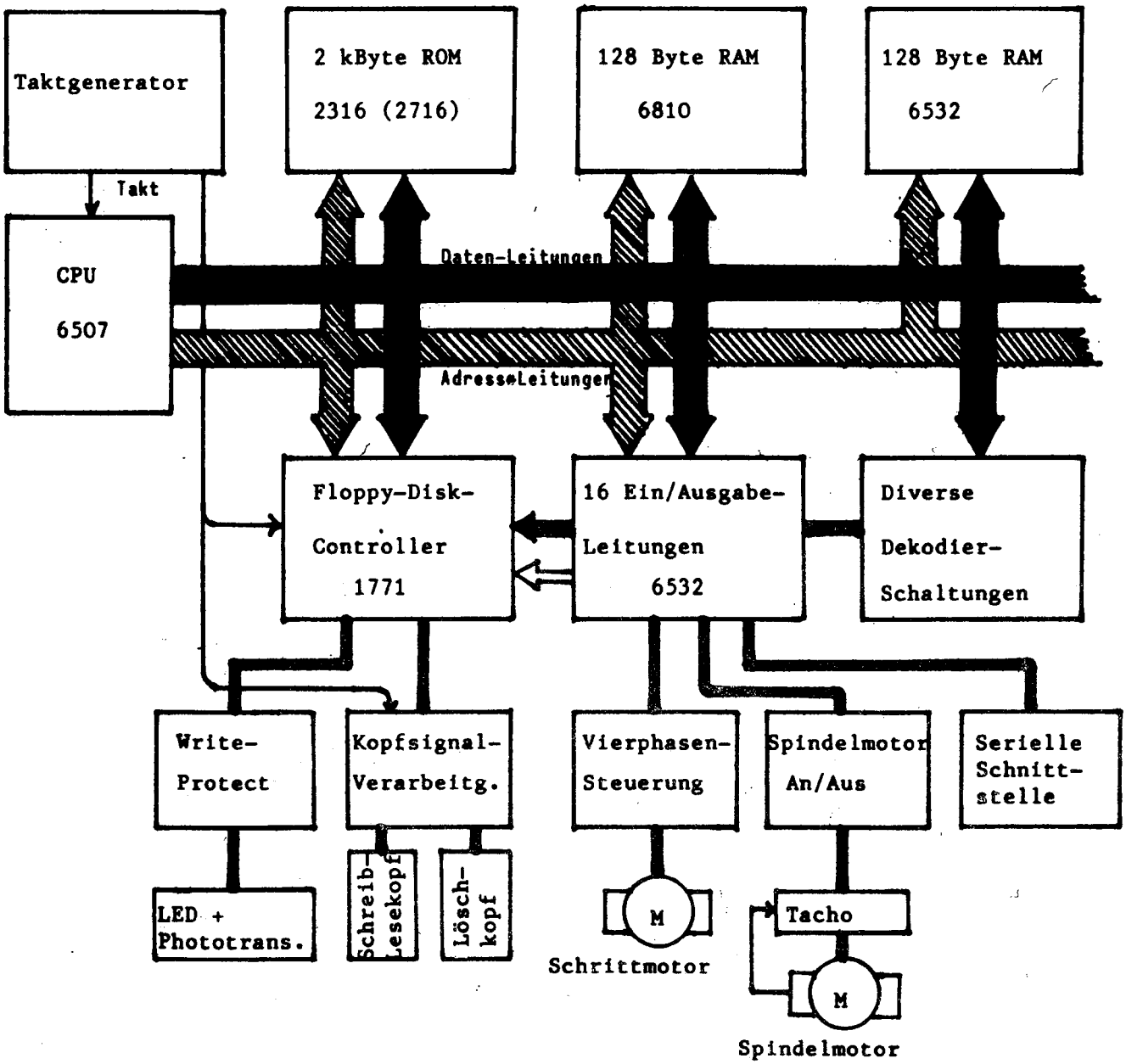


Abbildung 9 Vereinfachtes Prinzip-Schaltbild der Disk ATARI 810

Speicherkarte (Memory Map) der Diskettenstation ATARI 810

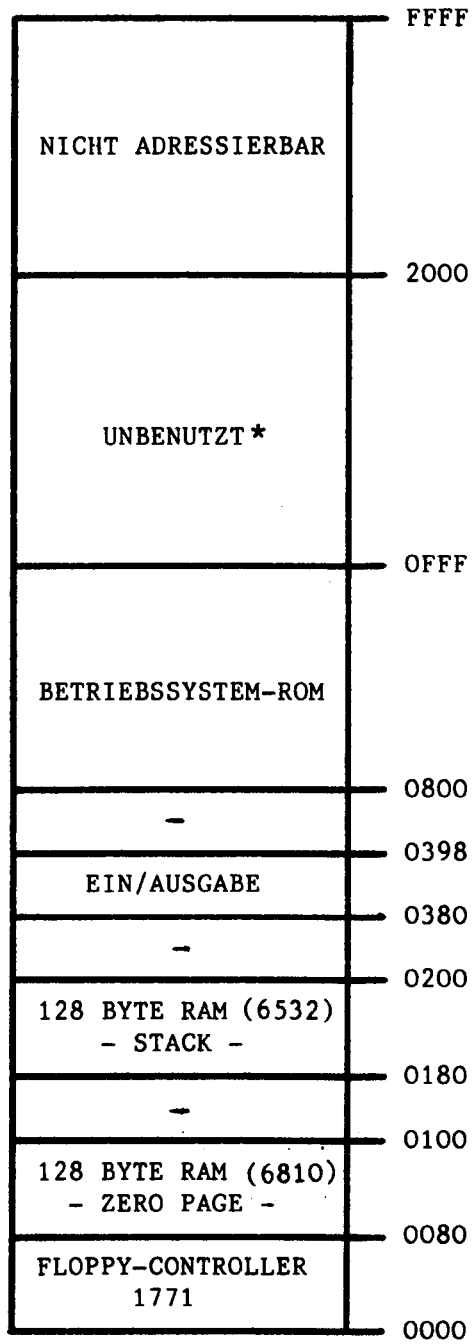
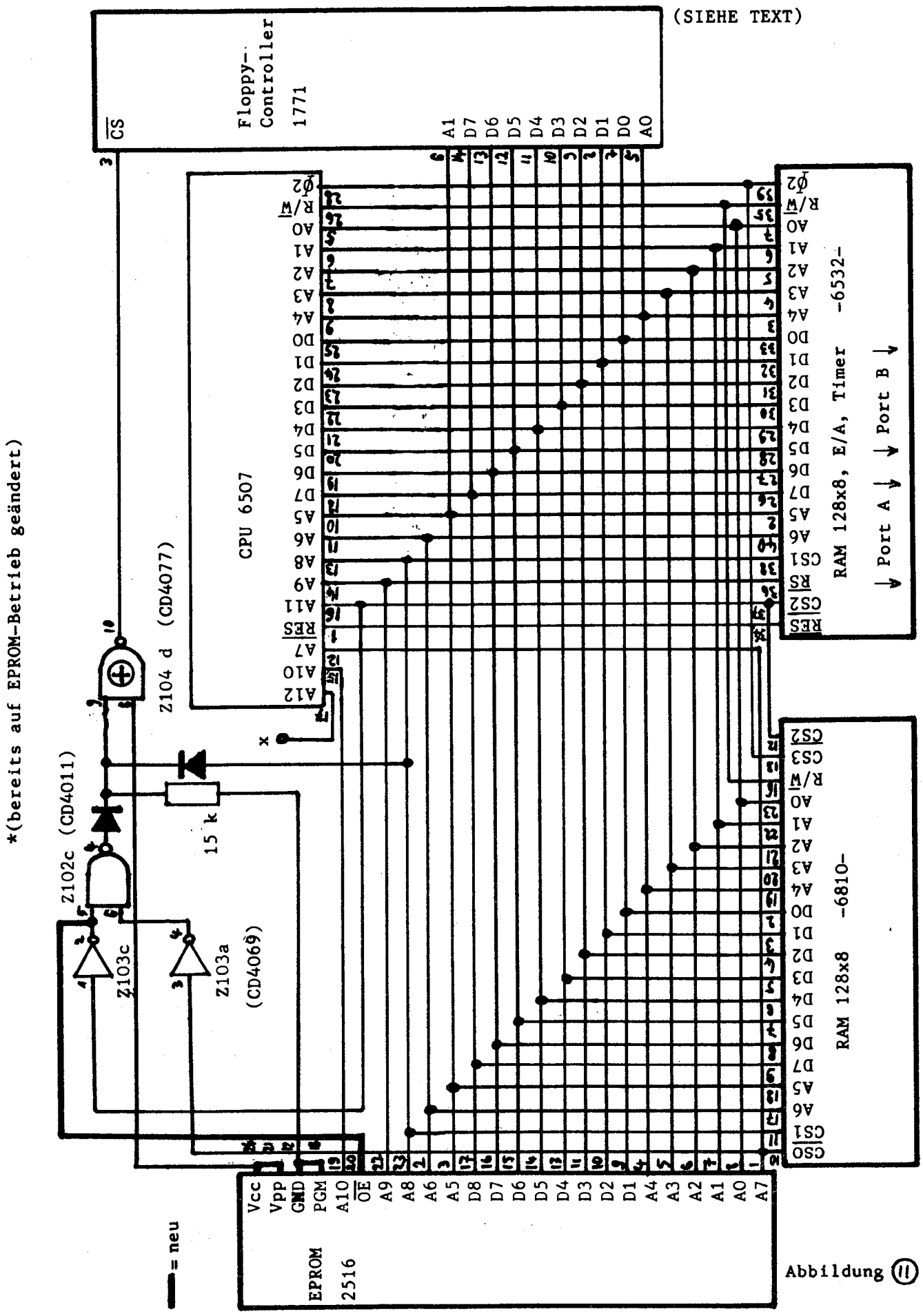


Abbildung 10

* Denkanstoß für Hardware/Maschninensprache-Freaks, die ihr Disk-Betriebssystem erweitern wollen: Durch Verbindung der bisher unbenutzten Adreßleitung A12 der 6507-CPU (siehe 'x' im Schaltplan) mit dem entsprechenden Anschluß eines 2532-EPROMs müßte es möglich sein, durch Einsatz eines 4k-EPROMs 2 kByte zusätzliche Programmspeicherkapazität zu bekommen. Man sollte jedoch die etwas unterschiedliche Anschlußbelegung vom 2532 gegenüber dem 2516 beachten (siehe Abschnitt 'Anschlußbelegung von ROMs und EPROMs').

SCHALTPLANAUSZUG: DER COMPUTER IN DER DISKETTENSTATION ATARI 810*



*(bereits auf EPROM-Betrieb geändert)

Abbildung 11

HARDWAREZUSATZ ZUM BESCHREIBEN DER DISKETTEN-RÜCKSEITEN

Es gibt grundsätzlich zwei Möglichkeiten, die Rückseiten normalerweise nur einseitig verwendbarer Disketten auszunutzen. Den Trick mit dem Ausschneiden eines Schreibschutzschlitzes auf der dem fabrikmäßig gestanzten Schlitz gegenüberliegenden Seite finden Sie im Kapitel 'DOS-TIPS'. Insbesondere für ATARI-810-Besitzer, deren Garantiezeit schon abgelaufen ist, bietet sich jedoch eine noch elegantere Methode an.

Normalerweise verhindert eine Lichtschranke in der linken Führungsschiene des Diskettenfachs das Formatieren und Bespielen einer Diskette, wenn der Schreibschutzschlitz überklebt ist. Die Rückseite ist daher mangels eines entsprechenden Schlitzes unzugänglich. Der ganze Trick besteht nun darin, die Lichtschranke an geeigneter Stelle zu überbrücken. Wir tun dies, indem wir den unteren Teil der Lichtschranke bei Bedarf einfach kurzschließen (siehe Pfeil in Abbildung 12).

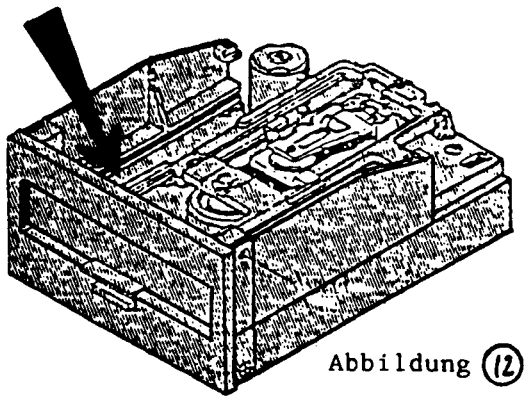


Abbildung 12

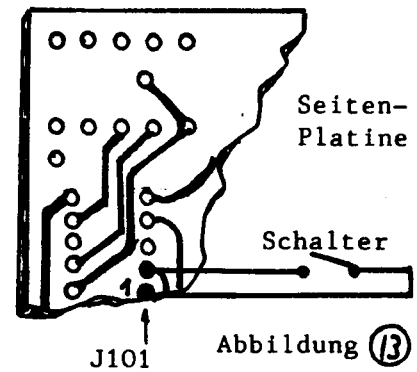


Abbildung 13

Wenn Sie die beiden Leitungen - ein rotes und ein schwarzes Kabel - zurückverfolgen, stellen Sie fest, daß sie in dem fünfpoligen Steckverbinder J101 hinten auf der Seitenplatine enden, und zwar als Pin 1 und Pin 2. Die entsprechenden Lötpunkte auf der anderen Seite der Platine warten direkt darauf, daß Sie dort je ein Kabel anlöten, um es mit einem Kurzschlußschalter zu verbinden (Abbildung 13). Am Besten besorgen Sie sich einen Schalter mit Zentrallochbefestigung (z.B. C K), der sich ohne großen Aufwand in die Gehäuse-Rückseite einbauen läßt.

Wenn Sie einen 2xEIN- oder 2xUM-Schalter verwenden, können Sie außerdem eine Leuchtdiode vorsehen, die Sie vor unbeabsichtigtem Löschen schreibgeschützter Disketten schützt (siehe Abbildung 14). Entfernt man auf der 810-Frontplatte links oben das aufgeklebte ATARI-Zeichen, kann man die LED dort gut sichtbar einbauen. Achtung: Um Verunreinigung zu vermeiden, sollten Sie die Laufwerkmechanik vor dem Bohren der Befestigungslöcher gut abdecken und hinterher Staub und Späne gründlich entfernen.

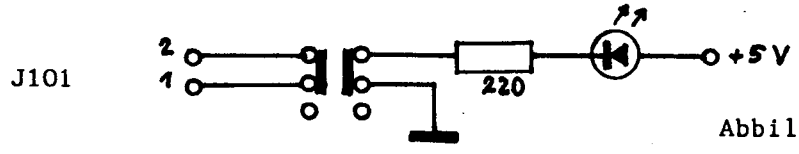
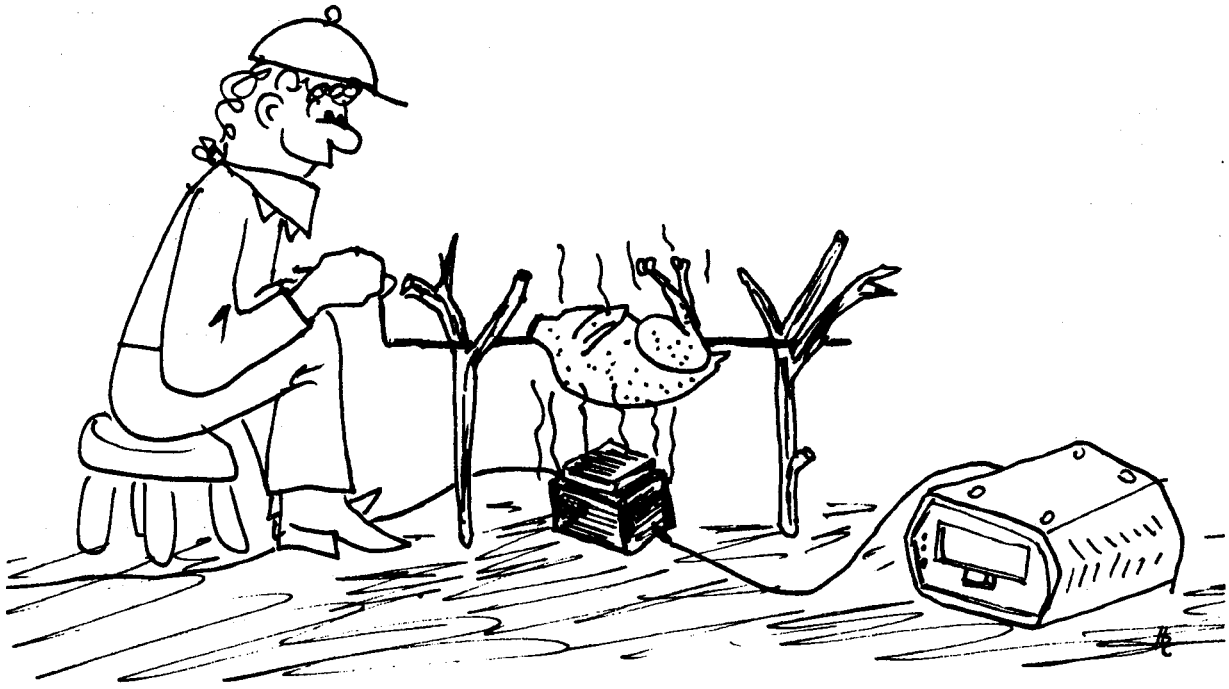


Abbildung 14

NETZTEILPROBLEME BEI ATARI 810

Wer einmal an der Sekundärwicklung des Transformators die Stromaufnahme seiner Diskettenstation ATARI 810 nachgemessen hat, wird mit Erstaunen einen Wert von ca. 2 Ampere abgelesen haben. Das ist für das mitgelieferte Netzteil (Typenschild: 9V/1,66A) auf Dauer eine sehr starke Belastung. Es ist vorstellbar, daß sich dies insbesondere bei Langzeitbetrieb nicht gerade stabilitätsfördernd auswirkt. Der Grund für diese Ersparnis am falschen Platz liegt wahrscheinlich in der ansonsten äußerst lobenswerten Idee, für alle Komponenten des ATARI-Systems einunddenselben Trafo zu verwenden, um tragischen Verwechslungen vorzubeugen. Wahrscheinlich haben diverse Modifikationen der ATARI 810, insbesondere die Ergänzung eines Datenseparators, zur nicht eingeplanten Erhöhung der Stromaufnahme geführt.

Wer mit seiner Diskettenstion auf der sicheren Seite arbeiten will, braucht also einen stärkeren Transformator. Es bieten sich zwei Wege an:



1. Man besorge sich den Trafo für den Commodore VC20 (!). Er ist mit 9V/3A spezifiziert und besitzt den gleichen Anschlußstecker wie der ATARI-Trafo, kommt als Ersatzteil jedoch ganz schön teuer.

2. Billiger ist es, sich in einem Bastel-Shop einen Transformator zu kaufen und ihn selber in ein Gehäuse zu stecken. Um für spätere Aufrüstungen gewappnet zu sein, sollten Sie sich für eine Type mit 3 Ampere oder mehr entscheiden, ansonsten reichen 2,5 A. Auch dem Anschlußkabel zur Diskettenstation schadet es nicht,wenn es etwas stärker ausfällt als beim Original-Netzteil. Nehmen Sie z.B. 2x1,75 mm^2 oder2x2 mm^2 Querschnitt.

Den 1,66A-Transformator sollten Sie übrigens unbedingt aufheben. Es kann ja sein, daß Sie eines Tages irgendein Zubehörgerät direkt aus USA ordern; mit dem meistens beiliegenden 110-Volt-Trafo können Sie nicht viel anfangen - und vielleicht paßt dann der 'alte' Trafo von der Disk 810.

Ein paar Hinweise zum Thema Stromversorgung

1. In jedem Fall ist es günstig, eine Schmelzsicherung in die Sekundärleitungen der Trafos einzubauen. Geeignete Sicherungshalter für einfache Kabelmontage kann man als Autoradio-Zubehör kaufen. Der Sicherungswert richtet sich nach der Stromaufnahme des Gerätes und ist entsprechend höher auszulegen. Für die Diskettenstation empfiehlt sich wegen der Motor-Stromspitzen eine träge Sicherung (T), ansonsten sollte man mittelträge (M) verwenden.

2. Für jedes ATARI-Gerät sollten Sie stets einen separaten Trafo oder eine galvanisch getrennte Sekundärwicklung verwenden.

3. Nur Trafos mit einer Leerlaufspannung von höchstens ca. 10 Volt verwenden (12 Volt sind ZUVIEL!).

4. Achten Sie bei allen Selbstbauten auf die elektrische Sicherheit, insbesondere auf ausreichenden Schutz vor Berührung und Kurzschluß!!

5. Das Kaltstarten ('Booten') geschieht beim ATARI 400/800 durch Einschalten des Computers. Da die Netzteil-Kondensatoren beim ausgeschaltetem Gerät noch eine Weile aufgeladen bleiben, sollte man zwischen Ausschalten und erneutem Einschalten stets ein paar Sekunden warten. Andernfalls kann es zu hohen Einschaltspitzen kommen, die insbesondere für dynamische RAMs gefährlich sind.

6. Bei längerer Abwesenheit sollten Sie die Netzstecker aller Transformatoren ziehen!

7. Von der Firma Kopp gibt es für ca. DM 18,- Blitzschutzstecker, die zwischen Steckdose und Verbraucher gesteckt werden. Es gibt zwei Ausführungen, und zwar für einen Schuko-Stecker oder für zwei flache Europa-Stecker. Man erkennt sie an ihrer grünen Farbe. Eigentlich für Fernseher etc. auf den Markt gebracht, müßten sie auch für das Computer-System Schutz gegen schädliche Überspannungen bieten.

DISK TEIL4: WARTUNG DER DISKETTENSTATION ATARI 810

Es gibt natürlich viele Service-Maßnahmen, die nur von geschultem Personal in gut ausgerüsteten Labors durchgeführt werden können, wie z.B. die Kopf-Justierung. Vieles, was die Lebenserwartung unseres Laufwerks und die Datensicherheit erhöht, können Sie jedoch auch selbst erledigen. Und eine Menge Geld dabei sparen. Allerdings sei wieder einmal daran erinnert, daß Sie mit dem Öffnen des Gerätes die Garantie aus Spiel setzen. Ist sowieso schon abgelaufen? Dann können wir ja gleich anfangen.

1. SCHMIERUNG

Der Geräuschpegel fabrikfrischer 810-Laufwerke läßt den Verdacht aufkommen, daß sich auf den mechanisch bewegten Teilen nicht ein Milligramm Schmiermittel befindet. Ein Blick ins Innere der Maschine bestätigt diesen Eindruck. Also frisch ans Werk: Alles, was Sie brauchen, sind Wattestäbchen, Pfeifenreiniger, ein paar Tropfen Nähmaschinenöl und etwas Silikonschmierstoff (in der Tube als Autozubehör erhältlich). Fett darf zum Schmieren des Laufwerkes nicht verwendet werden, da es sich bei Erwärmung verflüssigen und ungewollte Stellen benetzen kann. Auch Sprays sind ungeeignet, da ihre Anwendung meist mehr Schaden als Nutzen einbringt.

Schließen Sie das Diskettenfach und bewegen Sie die Kopf-Mechanik hin und her. Wie Sie sehen, läuft sie auf zwei runden Stahlschienen (siehe auch (a) Abb. (15)). Ungeschmiert sind sie die Hauptursache für die Geräschentwicklung des Laufwerks. Tragen Sie daher auf der Oberseite der beiden Schienen mit einem Wattestäbchen eine kleine Menge Silikon-Schmierstoff auf. Die Unterseite ist etwas schwer zugänglich und läßt sich am Besten mit einem entsprechend zurechtgebogenen Pfeifenreiniger schmieren. Die Geräuschminderung ist natürlich nicht der einzige Grund für die Schmierung der Schienen. Eine schwergängige Kopfführung bewirkt Schreib- und Lesefehler und sorgt für eine unnötige Belastung des Schrittmotors, was seiner Lebensdauer nicht gerade zuträglich ist.

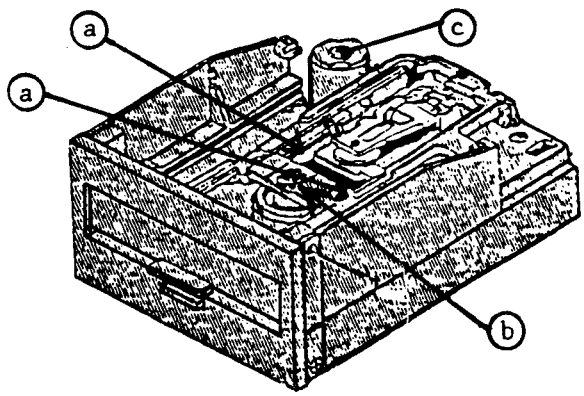


Abbildung (15)

Schmierbedürftig sind auch die Lager der Spindelführung (b) und des Spindelmotors (c). Versorgen Sie diese beiden Stellen mit einem k-l-e-i-n-e-n Tröpfchen Öl. Alle anderen mechanischen Teile sollten Sie ungeschmiert belassen. Lösen Sie auf keinen Fall irgendwelche Schrauben, die direkt mit der Mechanik zu tun haben. Vieles ist hier werkseitig genauestens justiert! Bewegen Sie den Kopf-Mechanismus einige Male hin und

her, um den Schmierstoff zu verteilen und entfernen Sie überschüssiges Schmiermittel mit einem Wattestäbchen. Wenn Sie Ihre Diskettenstation nun ausprobieren, werden Sie staunen: selbst bei geöffnetem Gehäuse sind Formatieren, Kopfverschiebung und Spindelrotation akustisch kaum noch wahrnehmbar!

2. KOPF-REINIGUNG

Wie Sie inzwischen bereits gesehen haben, ist die in Abbildung gezeigte Kombination aus Kopf und Andruckfilz auf zwei Rundschienen gelagert. Mit Hilfe eines flexiblen Metallbandes kann sie vom rechts daneben befindlichen Schrittmotor in 40 genau festgelegten Schritten verschoben werden. Die Steuer-Elektronik zählt stets mit, auf welcher Spur sich der Kopf befindet und benötigt nur Millisekunden, um ihn zu irgendeiner anderen Spur zu fahren. Der Kopf hat dabei fortwährend Kontakt mit der Disketten-Oberfläche und neigt aufgrund der Reibung genau wie der Andruckfilz zur Verschmutzung. Beide Teile bedürfen je nach Benutzungshäufigkeit alle drei bis sechs Monate einer gründlichen Säuberung. Auch dafür benötigen Sie ein paar Wattestäbchen, außerdem etwas denaturierten Alkohol aus der Apotheke. Tränken Sie ein Stäbchen mit Alkohol und reinigen Sie vorsichtig den Filz. Nehmen Sie dann ein frisches Wattestäbchen und tupfen Sie den Filz wieder trocken.

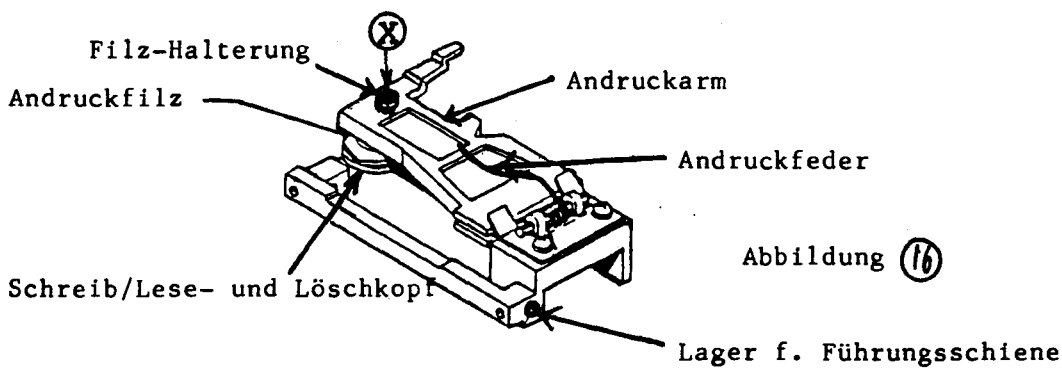


Abbildung 16

ÜBRIGENS... Bei Formatierungsproblemen kann es unter Umständen aus unerfindlichem Grund hilfreich sein, wenn man die Orientierung der Filzfasern etwas ändert. Verdrehen Sie dazu mit einem Schraubenzieher den Schlitz in der Filz-Halterung (x in Abbildung 16). Durch Ausprobieren findet man oft eine günstigere Stellung. Notieren Sie sich aber am Besten zuvor die Ausgangsstellung.

Auch der Kopf selbst wird mit Wattestäbchen und (wenig) Alkohol gereinigt. Seine Oberfläche sollte möglichst nicht mit den Fingern berührt werden. WICHTIG: Achten Sie bei Ihrer Reinigungsaktion darauf, daß Sie beim Anheben des Andruckarms die Feder nicht überdehnen. Damit wären wir auch schon beim nächsten Punkt:

3. FEDERDRUCK-KONTROLLE

Wer eine feine Federwaage sein eigen nennt, kann auch die Einstellung des Andruckarms überprüfen. Bei geschlossenem Diskettenfach sollte der Filz mit 20 bis 30 Gramm gegen den Kopf drücken. Andernfalls muß die Andruckfeder korrigiert werden. Während ein zu hoher Federdruck Kopf und Diskette frühzeitig verschleißen läßt, führt eine zu schwach eingestellte Feder zu Fehlern beim Lesen, Schreiben und Formatieren. Siehe auch '6. VORBEUGENDE WARTUNG'.

4. DREHZAHL-EINSTELLUNG

Auch wenn Ihre Diskettenstation bereits über einen Datenseparator verfügt, der ja die Datensicherheit von Geschwindigkeitsabweichungen weitgehend unabhängig macht, sollte die Nenndrehzahl von 288 Umdrehungen pro Minute (Upm) zumindest einigermaßen genau eingehalten werden (ca. +/-3 Upm). Dafür gibt es gute Gründe:

1. Wenn die Drehzahl zu hoch ist:
Dreht sich die Spindel zu schnell, ist dies eine häufige Ursache für Formatierungsfehler. Denn bevor der letzte (also der 18.) Sektor einer Spur fertiggeschrieben wurde, hat die Diskette bereits eine vollständige Umdrehung absoviert, sodaß der Anfang des ersten Sektors überschrieben wird. Meistens erscheint dann nach einigen vergeblichen Anläufen die Fehlermeldung 'ERROR173' (Formatierungsfehler). Sollte das Formatieren gerade noch möglich sein, haben die Besitzer anderer Diskettenstationen mit Sicherheit Schwierigkeiten beim Lesen Ihrer Disketten. Und wenn Sie Disketten bekommen, die auf einer langsameren Maschine formatiert wurden, haben Sie wiederum Schwierigkeiten beim Abspeichern, da Sie über das tatsächliche Ende der Sektoren hinausschreiben.

2. Wenn die Drehzahl zu niedrig ist:
Dreht sich die Spindel zu langsam, werden die Daten so eng gepackt, daß sie schwierig oder überhaupt nicht gelesen werden können, insbesondere von schnelleren (normal eingestellten) Laufwerken. Da man mit zu langsamen Diskettenstationen zwar mühelos zu schnell bespielte Disketten laden kann, aber nicht umgekehrt, kommt man diesem Fehler kaum auf die Schliche. Besonders die inneren Spuren mit den höheren Sektor-Nummern sorgen immer wieder für Verdruß, da die Daten hier enger gepackt sind als außen. Dies macht sich um so mehr bemerkbar, je niedriger die Drehzahl eingestellt ist.

Ist eine Korrektur der Drehzahl erforderlich, geschieht dies durch Einstellen am Trimm-Potentiometer, daß sich auf der Bodenplatine hinten rechts befindet (Siehe Abbildung 17). Drehen Sie dieses Poti mit einem Schraubenzieher nach links, erhöht sich die Drehzahl. Drehen Sie nach rechts, wird der Motor lansamer.

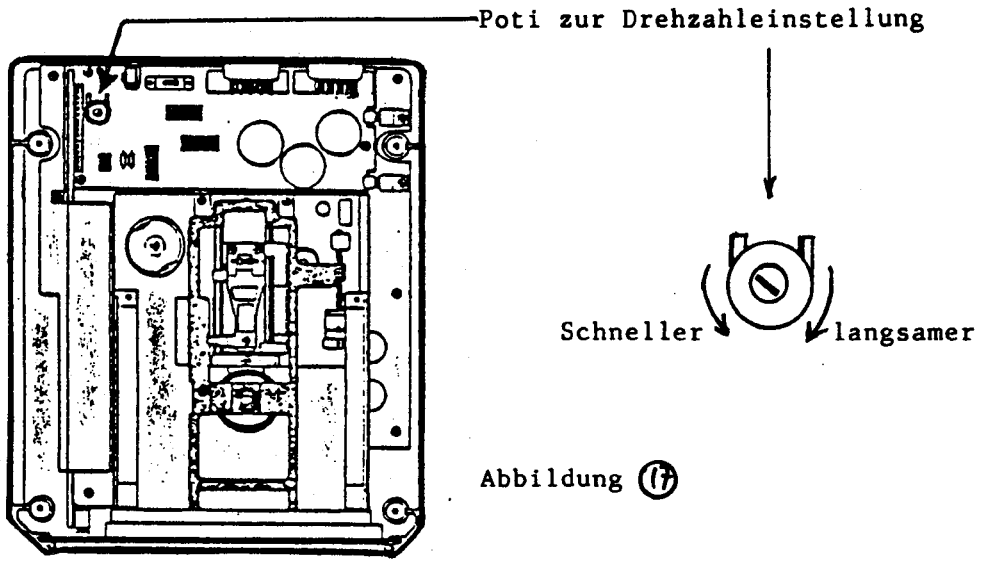


Abbildung 17

Wie aber können wir die Drehzahl selber ermitteln, um sie gegebenenfalls zu korrigieren?

Schnell und komfortabel geht's mit dem nachfolgenden BASIC-Programm, daß Ihnen auf +/-1 Upm genau sagt, wie schnell bzw. langsam Ihr Laufwerk ist. Alles, was Sie tun müssen, ist:

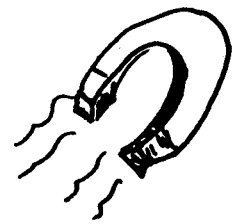
1. Das Programm eintippen (und abspeichern)
2. Eine formatierte Diskette einlegen
3. RUN eingeben.

Alles andere sagt Ihnen das Programm selbst.

```
100 REM DREHZAHL-TEST
110 FOR A=0 TO 74:READ B:POKE 1536+A,B:NEXT A
120 ? "":SETCOLOR 2,2,10:SETCOLOR 1,0,0:POKE 752,1:POSITION 4,10:?
"LAUFWERK-NUMMER ? ";
130 OPEN  1,4,0,"K:":GET  1,LN:CLOSE  1:? LN-48:POKE 1617,LN-48
140 ? :? "  BITTE ETWAS GEDULD ...":X=USR(1536)
150 Z1=PEEK(1618):Z2=PEEK(1619)
160 MIN=(256 Z2+Z1)/3600
170 UPM=INT((100/MIN+0.5) 5/6)
180 ? :? :? "  DREHZAHL = ";UPM;" UPM"
190 IF UPM>284 AND UPM>292 THEN ? :? "DAS IST OK!":END
200 IF UPM<285 THEN ? :? "  DAS IST ZU NIEDRIG!":? :? "  POTI NACH LINKS
DREHEN!":END
210 IF UPM>291 THEN ? :? "  DAS IST ZU HOCH!":? :? "  POTI NACH RECHTS
DREHEN!":END
220 DATA 104,169,1,141,10,3,169,0,141,11,3,169,128,141,4,3,169,6
230 DATA 141,5,3,173,81,6,141,1,3,169,82,141,2,3,169,5,141,80
240 DATA 6,32,83,228,206,80,6,208,248,169,100,141,80,6,169,0,133,19
250 DATA
133,20,32,83,228,206,80,6,208,248,165,20,164,19,141,82,6,140,83,6,96
```

5. KOPF-ENTMAGNETISIERUNG

Je nach Benutzungshäufigkeit sollte der Schreib- Lesekopf in viertel- bis halbjährlichen Intervallen entmagnetisiert werden, da sich im Laufe der Zeit ein Magnetfeld aufbaut, was zu Lesefehlern führen kann. Sehr gut eignen sich dazu Entmagnetisier-Drosseln, wie sie für ca. DM 20,- zur Behandlung von Tonband- und Cassettenrecorder-Tonköpfen angeboten werden. Wichtig ist, daß Sie die Drossel ganz langsam und in großem Bogen wieder vom Kopf entfernen, damit nicht eines der 50-mal pro Sekunde wechselnden Magnetfelder zurückbleibt. Meistens wird diese Prozedur auch in der Gebrauchsanleitung der Drossel eingehend erläutert. - Vorteilhaft, wenn auch nicht unbedingt erforderlich ist es übrigens auch, wenn Sie für alle Arbeiten an Ihrem Laufwerk nichtmagnetische Schraubenzieher verwenden.



6. VORBEUGENDE WARTUNG:

Ein paar gutgemeinte Tips zum Umgang mit Disketten und Laufwerk

1. Was die Handhabung der Disketten betrifft, dürfen Sie damit alles Mögliche machen, z.B. das Diskettenfach öffnen, obwohl die BUSY-Lampe leuchtet, Diskette rein/Diskette raus, Disk ausschalten etc., NUR DAS EINE NICHT: das Laufwerk EINschalten, wenn eine Diskette drinsteckt. Andernfalls ist die Chance, daß dann ein paar Bits draufgeschrieben werden, bevor alle Logikpegel in der Steuer-Elektronik einen stabilen Zustand angenommen haben, recht groß.

2. Lassen Sie das Diskettenfach nicht auf Knopfdruck aufspringen, sondern beremsen Sie die Aufwärtsbewegung der Klappe mit den Fingern ab. Andernfalls kann es mit der Zeit zu einer Überdehnung der Feder kommen, sodaß der Andruckfilz dann nicht mehr stark genug gegen Diskette und Kopf drückt. Ein häufiger Grund für Formatierungs- Schreib- und Lesefehler.

3. Wer seine 'nackten' Disketten mit dem Etikett nach oben auf den Tisch legt, ist sich anscheinend nicht darüber im Klaren, daß der Kopf von unten zugeführt wird. Man riskiert also eine Gefährdung der wichtigen Datenseite durch Staub, Schmutz und Kratzer. Nach Benutzung sollten Sie Ihre Disketten stets sofort in die dazugehörigen Papiertaschen zurückstecken (Merke: man muß auch mal zurückstecken können!).

4. Eine Diskette ist schnell 'gestorben', wenn sie auf dem Arbeitstisch versehentlich auf Scheren, Kugelschreiber, Schraubenzieher o.ä. gelegt wird. Alle diese Gegenstände können magnetisch sein und die empfindlichen Datenstrukturen auf der Diskette zerstören. Meistens liest man dann nur noch ERROR144 oder ERROR164.

5. Machen Sie von allen wichtigen Disketten SOFORT nach Erhalt bzw. Fertigstellung Sicherheitskopien (auf neudeutsch: Back-Ups) !!! Das ist in jedem Falle billiger als neu kaufen oder neu tippen.

6. Transportsicherung: Wenn Sie Ihre Diskettenstation im Auto transportieren, sollten Sie stets die Pappscheibe in den Diskettenschacht stecken, die sich bei Lieferung Ihres Laufwerks darin befand. Andernfalls kann es durch Erschütterungen zu Beschädigungen oder zum Verstellen der Justierung des Kopfes kommen. Sollte die Pappe bereits im Müll gelandet sein, können Sie stattdessen auch eine (alte) Diskette verwenden. - Auch wenn Sie normalerweise alle Kartons wegwerfen: die 810-Originalverpackung sollten Sie unbedingt aufheben, da sie bei größeren Transporten optimalen Schutz vor Erschütterungen bietet.

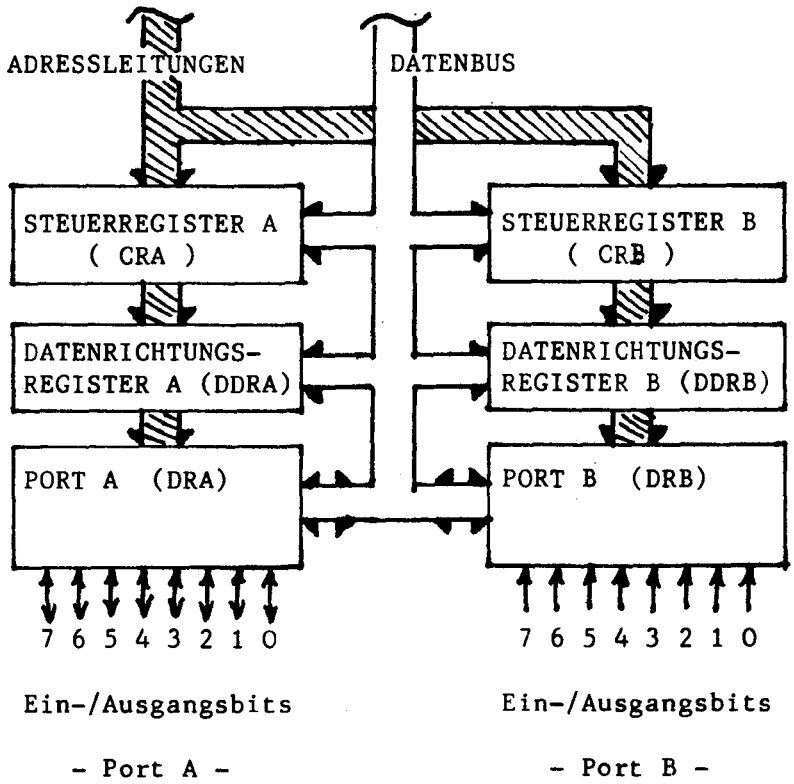
7. Außer pfleglicher Behandlung bedürfen die Disketten selbst übrigens keiner Wartung. Ihren Reinigungsmechanismus haben sie herstellerseitig bereits eingebaut bekommen: er besteht aus der weichen Fütterung des sogenannten Jacketts und dem Gleitmittel, mit dem die Auskleidung getränkt ist.

SCHNITTSTELLEN-PROGRAMMIERUNG TEIL 1: DIE STEUERKNÜPPEL-ANSCHLÜSSE

Die verschiedenen Zusatzgeräte, die es mittlerweile für den ATARI gibt, lassen schon ahnen: an die vier Buchsen vorne unter der Tastatur lassen sich weit Dinge mehr anschließen als nur Steuerknüppel!

So werden mittlerweile EPROM-Programmiergeräte, Drucker- und Harddiskschnittstellen, Antennenrotor-Steuerungen etc. angeboten, die über die Steuerknüppel-Buchsen mit dem Computer verbunden werden und deren wichtigste Eigenschaft ausnutzen: die PROGRAMMIERBARKEIT. Jede der vier Buchsen ist (unter anderem) mit vier Schnittstellenleitungen belegt, die sich in beliebiger Kombination einzeln wahlweise als TTL-kompatibler Eingang oder Ausgang programmieren lassen. Zur Steuerung dieser 16 Leitungen befindet sich im ATARI der Schnittstellenbaustein 6520 ('PIA', Peripheral Interface Adapter). Er enthält neben zwei programmierbaren bidirektionalen 8Bit-Ports auch noch vier Steuerleitungen, die jedoch nur intern Verwendung finden.

Da sich mit Hilfe der BASIC-Befehle STRIG, STICK(0) usw. die vier Buchsen nur als Eingange abfragen lassen, ist eine individuelle Programmierung der beiden Ports nur mit PEEKs und POKEs bzw. in Maschinensprache möglich. Dazu ist es wichtig, den Aufbau der PIA 6520 zu kennen (Abb. 18).



19

Wie Abb.. zeigt, besteht der PIA-Baustein aus zwei identischen Blöcken, von denen sich jeder aus drei Registern zusammensetzt:

1. Steuerregister CR
2. Datenrichtungsregister DDR
3. Ein-/Ausgabe-Datenregister (Port) DR

Wie oben erwähnt läßt sich bitweise festlegen, welche Leitungen der beiden Ports als Ausgang und welche als Eingang arbeiten sollen. Dies geschieht mit Hilfe der Datenrichtungsregisters (DDRA bzw. DDRB). Nullen legen in diesem Register fest, daß die entsprechende Bits der Ports (DRA bzw. DRB) als Eingänge geschaltet sind. Dies ist z.B. bei der Abfrage der Steuerknüppel der Fall. Demgegenüber sind alle Port-Leitungen als Ausgänge programmiert, deren zugehörige Steuerregister-Bits Einsen enthalten. Noch unklar? Hier ein Beispiel:

```
Bit Nr.   7 6 5 4 3 2 1 0
DDRA      0 0 0 0 1 1 1 1
```

Indem wir in das Datenrichtungsregister von Port A den Wert OF (=0000 1111) schreiben, haben wir die vier höherwertigen Bits als Eingänge und die vier niederwertigen Bits als Ausgänge definiert. Dieser Zustand bleibt solange erhalten, bis das Programm bzw. ein Poke-Befehl eine Umdefinition vornimmt (oder jemand unseren ATARI ausschaltet).

Die Datenrichtungsregister sind also über Adressen zu erreichen. Die Ports und Steuerregister auch. Doch nun zu einer Besonderheit der PIA 6520: um Adreßleitungen und Speicherplatz zu sparen, HAT DER PORT JEWEILS DIESELBE ADRESSE WIE SEIN DATENRICHTUNGSREGISTER! Wie das möglich ist? Durch einfaches Umschalten. Der 'Schalter' dazu befindet sich im jeweiligen Steuerregister (CR): Bit 2 (das dritte Bit von rechts) gibt an, auf welches Register die Weiche gestellt ist. Enthält dieses Bit den Wert 0, wird das Richtungsregister angesprochen. Enthält es eine Eins, erreicht man unter derselben Adresse den Port. Dazu wieder ein Beispiel:

```
Bit Nr. 7 6 5 4 3 2 1 0
CRB     0 0 0 0 0 X 0 0

X=0 : $D301 ist die Adresse von DDRB
X=1 : $D301 ist die Adresse von DRB
```

Wie aus Tabelle .. zu entnehmen, hat das Steuerregister A die Adresse $D303 und das Datenrichtungsregister A bzw. Port A die Adresse $D301. Weist man $D303 den Wert $00 zu, erreicht man unter $D301 das Datenrichtungsregister B und kann dort bitweise die Eingänge bzw. Ausgänge festlegen. Schreibt man dann in die Zelle $D303 den Wert $04 (= 0000 0100), spricht man unter der Adresse D301 an Port B angelegte Signale einlesen bzw. Bitmuster dorthin aussenden. In BASIC sieht das folgendermaßen aus:

```
10 REM PORT B ALS EINGANG
20 POKE 54019,48:REM DDRB ANWAEHLEN
30 POKE 54017,0:REM ALLE BITS = EING.
40 POKE 54019,52:REM PORT B ANWAEHLEN
50 PRINT PEEK(54017): REM PORT B LESEN

10 REM PORT B ALS AUSGANG
20 POKE 54019,48:REM DDRB ANWAEHLEN
30 POKE 54017,255:REM ALLE BITS = AUSG.
40 POKE 54019,52:REM PORT B ANWAEHLEN
50 POKE 54017,N:REM BITMUSTER N AUSGEBEN
```

WICHTIG: Es gilt zu beachten, daß beim Setzen der Steuerregister anstelle von $00 und $04 die Werte $30 bzw. $34 zu verwenden sind (dezimal 48 bzw. 52). Der Grund dafür liegt darin, daß andere Steuerregister-Bits für die

intern verwendeten PIA-Steuerleitungen zuständig sind. Um die korrekte Arbeitsweise des Betriebssystems sicherzustellen, muß der Zustand dieser Bits erhalten bleiben.

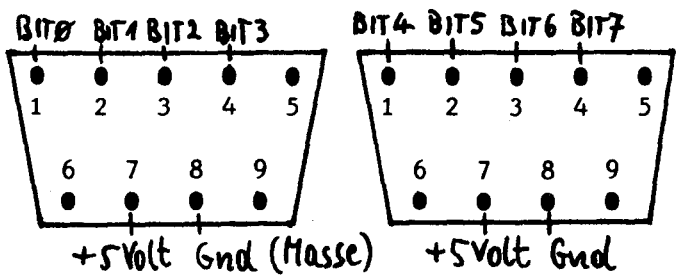
Dasselbe Beispiel in ASSEMBLER:

```
05 *=$0600
10 DDRB=$D301
15 CRB =$D303
20 ;
25 ;PORT B ALS EINGANG
30 LDA  $30
35 STA CRB; DDRB ANWAEHLEN
40 LDA  $00
45 STA DDRB; ALLE BITS = EING.
50 LDA  $34
55 STA CRB; PORT B ANWAEHLEN
60 LDA DDRB;PORT LESEN
65 ;
70 ;PORT B ALS AUSGANG
75 LDA  $30
80 STA CRB; DDRB ANWAEHLEN
85 LDA  $FF
90 STA DDRB; ALLE BITS = AUSG.
95 LDA  $34
96 STA CRB; PORT B ANWAEHLEN
97 LDA  $AA
98 STA DDRB; Z.B. DAS BITMUSTER $AA AUSSENDEN
99 .END
```

(Bei Maschinen-Unterprogrammen, die von BASIC aufgerufen werden sollen: PLA und RTS nicht vergessen !!! - Siehe ATARI BASIC Reference Manual)

Die Programmierung kann natürlich auch mit gemischten Ein- und Ausgängen erfolgen. Zum Lesen bzw. Schreiben der Ports muß dann softwaremäßig eine entsprechende Maskierung der Bytes vorgenommen werden. Zum LESEN der Ports läßt sich die Maskierung verwenden, die im Betriebssystem für die Abfrage der Steuerknüppel vorgesehen ist. So kann man z.B.in BASIC mit STICK(3) den Zustand der vier höherwertigen Bits von Port B ($D301) abfragen. Die entsprechenden Adressen finden Sie im Kapitel "DIE MEMORY MAP oder WAS LIEGT WO?".

Nachdem wir nun wissen, wie die Ports zu programmieren sind, interessiert uns deren Verdrahtung mit den Anschlußbuchsen: siehe Abb. 19 .



Port A: Buchse 1 u. 2
Port B: Buchse 3 u. 4

19

BUCHSE 1 (3) BUCHSE 2 (4)

Die Adressen der PIA-Register:

	Hex.	Dez.
DRA/DDRA	D300	54016
DRB/DDRB	D301	54017
CRA	D302	54018
CRB	D303	54019

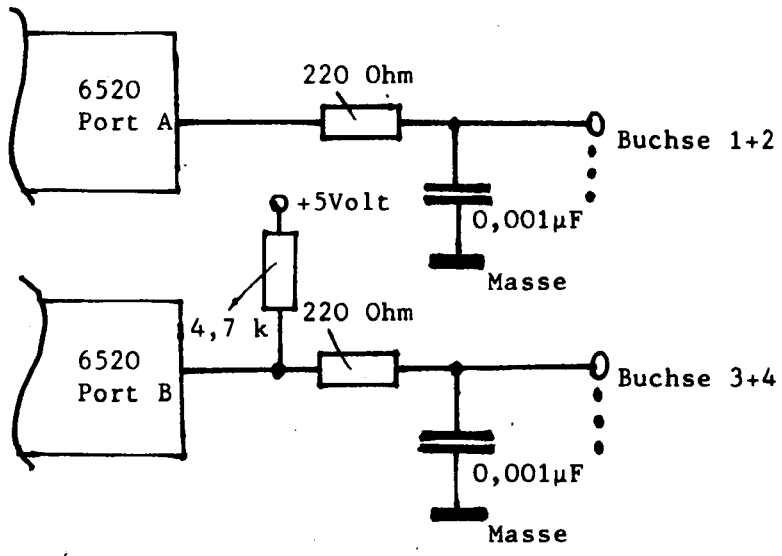
An anderer Stelle finden Sie in diesem Buch Bauanleitungen für Schnittstellen, die natürlich auch als Beispiele für eigene Entwicklungen dienen können.

ÜBRIGENS.... Die passenden Stecker finden Sie in diesem Buch unter 'STECKER UND KABEL'.

!!! WICHTIG !!! !!! WICHTIG !!!

Für den Anschluß eigener Hardwarezusätze an die Steuerknüppel-Buchsen ist unbedingt zu beachten, daß die interne 5 Volt-Stromversorgung des ATARI-Computers höchstens mit 50 mA belastet werden darf !!!!!!!!

Desweiteren unterscheiden sich Port A und Port B in ihrer internen und externen Beschaltung (intern: siehe Datenblatt 6520 v. Commodore, Synertek oder Rockwell - extern: siehe unten).

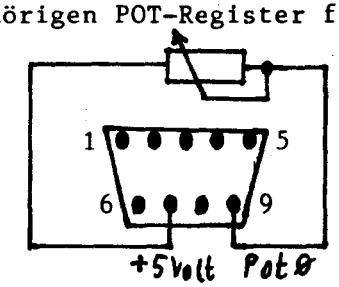


SCHNITTSTELLEN-PROGRAMMIERUNG TEIL 2: DIE DREHREGLER-ANSCHLÜSSE

Was die Besitzer der meisten anderen Computer teuer zukaufen müssen, ist im ATARI bereits ab Werk eingebaut: eine Analog-Schnittstelle nebst dazugehöriger Software! Neben den 16 digitalen Ein-/Ausgabeleitungen befinden sich an den Anschlußbuchsen auch die acht A/D-Wandler-Eingänge zur Abfrage der Drehregler (Paddles). Eigentlich zu deren Digitalisierung gedacht, damit Sie bei diversen Spielen einen flotten Schläger schwingen können, eignen sie sich auch für zahlreiche andere Anwendungen. Siehe unser Beispiel 'Helligkeitsregelung'. Um die Paddle-Ports optimal nutzen zu können, hier eine kurze Funktionsbeschreibung, gefolgt von ein paar kleinen Beispielen.

Als Eingangsgröße werden Widerstände gemessen. Denn in den Drehreglern steckt natürlich nichts anderes als ein regelbarer Widerstand (Potentiometer) von jeweils 1 MOhm. Im Inneren des ATARI ist mit jedem angeschlossenen Drehregler-Widerstand ein Kondensator in Serie geschaltet. Nach vorheriger Entladung über einen Transistor wird er im 50Hz-Takt aufgeladen - wie schnell dies passiert, hängt von der Größe des eingestellten Widerstands ab. Der in das entsprechende POT-Register geschriebene Wert zeigt an, wieviele Bildschirmzeilen es gedauert hat, den Kondensator bis zur vorgegebenen Schwellenspannung aufzuladen. Dieser Wert liegt - je nach Einstellung des Drehreglers - zwischen Null (Regler ganz nach rechts gedreht) und 228 (Regler am linken Anschlag). Aufgrund von Fertigungstoleranzen der Paddles kann der tatsächlich einstellbare Maximalwert jedoch auch unterhalb von 228 liegen. Je höher der Widerstand, desto größer ist die Zahl, die wir im dazugehörigen POT-Register finden.

Abb.20: Innenleben eines Drehreglers



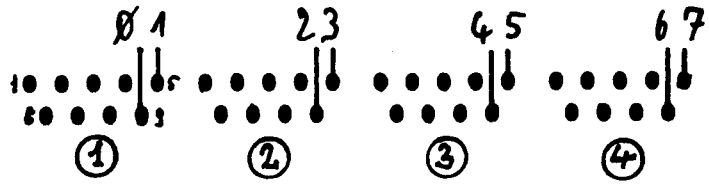
Die Abfrage der Zahlenwerte erfolgt entweder mit Hilfe der BASIC-Funktionen PADDLE(Ø) - PADDLE(7) oder durch direktes Lesen der Adressen von POTØ bis POT7:

	Hex.	Dez.
POTØ	$D200	53760
POT1	$D201	53761
POT2	$D202	53762
POT3	$D203	53763
POT4	$D204	53764
POT5	$D205	53765
POT6	$D206	53766
POT7	$D207	53767

Bei der Verwendung von Player/Missile-Objekten sollte man übrigens darauf achten, daß die POT-Werte nicht kleiner als 40 und nicht größer als 200 werden können, da sie andernfalls nicht mehr auf dem Bildschirm zu liegen kommen.

An den Anschlußbuchsen liegen die Paddle-Eingänge von links nach rechts in aufsteigender Reihenfolge (Abb.21):

PADDLE Nr...



21

Natürlich gibt es wesentlich genauere und höher auflösende Analog/Digital-Wandlungsvefahren, als das im ATARI verwendete Methode. Aber ganz abgesehen von Kosten/Nutzen-Überlegungen halte man sich vor Augen, daß beispielsweise auch das Grafik-Tablett VERSAWRITER (Hersteller: Versa Computing/USA) an die Drehregler-Eingänge angeschlossen wird. Dennoch bietet dieses einfache Gerät (2 Potentiometer/etwas Kunststoff - der Rest ist Software) eine erstaunliche Genauigkeit in der Reproduktion von Abbildungen aller Art.

Was man vom Design des VERSAWRITER lernen kann: wenn es bei Eigenentwicklungen ebenfalls auf Genauigkeit ankommt, ist wegen der relativ großen Toleranzen der Paddle-Eingänge unbedingt die Möglichkeit des Abgleichs auf einen Bezugspunkt vorzusehen.

WISSENSWERTES ÜBER EPROMS

WARUM BRAUCHE ICH EPROMS?

Für den Fall, daß Unklarheit darüber bestehen sollte, warum die EPROMs so nützliche Bausteine für Ihren ATARI-Computer sind, hier eine kurze Erläuterung.

Grundsätzlich unterscheidet man in einem Computer-System ja zwischen zwei Arten von Speichern:

1. R A M

Der Schreib- Lesespeicher (RAM) dient zur Aufnahme von Daten, die sich während des Programmablaufs ändern. Einen Teil des RAM verwaltet das System selbst, um für den Betrieb wichtige Daten abzulegen, wie z.B. Graphik-Modus, Timer, Tabellen, Stapelregister etc. Der Rest steht dem Anwender zur Verfügung, um alle möglichen daten abzulegen: Programmbefehle, Tabellen, Bildpunkte für graphische Darstellung auf dem Bildschirm etc. Eines haben alle RAM - Speicherzellen gemeinsam: beim Ausschalten des Computers geht ihr Inhalt verloren. Um sich das mühsame Neu-Eintippen von Programmen zu ersparen, wurde daher die Möglichkeit der Speicherung auf Cassette bzw. Diskette vorgesehen.

2. R O M

Der zweite Speicher-Typ zeichnet sich dadurch aus, daß seine Daten bei Ausschalten des Computers nicht verloren gehen: das ROM ist ein nichtflüchtiger Nur-Lese-Speicher (Read Only Memory). Sein Inhalt enes ROMs kann während des Betriebes nicht geändert werden; dafür steht er jedoch beim Einschalten des Computers sofort zur Verfügung. Als ROM ist z.B. das Betriebssytem (OS) Ihres ATARI-Computers gespeichert, und natürlich die zahlreichen Steckmodule mit BASIC, PILOT und den vielen spannenden Spielen. Wir sind es gewohnt, diese Module einfach in unseren Rechner hineinzustecken, einzuschalten und das jeweils enthaltene Programm sofort bertriebsbereit vor uns zu haben - ohne lange Ladezeiten.

WAS STECKT NUN IN SO EINEM ROM-MODUL ?

Zum besseren Verständnis sollten Sie sich die Mühe machen, ein ROM-Steckmodul aufzuschrauben. Sie brauchen nur die Kreuzschlitz-Schraube zu entfernen und die Blechkappe abzunehmen: Sie sehen nun eine kleine Platine. Neben einem Kondensator befinden sich darauf ein oder zwei 24-polige IC-Bausteine. Dies sind die ROMs. Ihre Anschlüsse teilen sich folgendermaßen auf:

1. Zwei Stromversorgungsleitungen
2. Acht Datenleitungen
3. 10 bzw. 11 Adressleitungen (je nach Speichergröße)
4. Der Rest wird für Auswahlleitungen verwendet

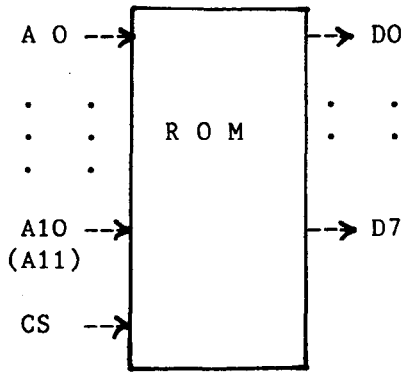


Abbildung 22
Funktionsprinzip eines ROMs

Mit Hilfe der 10 bzw. 11 Adressleitungen lassen sich im Binärsytem 2 = 2096 bzw. 2 = 4192 speicherzellen adressieren, also 2 kByte bzw. 4 kByte (1 kbyte = 1024 Bytes). Da z.B. das ATARI-BASIC-ROM zwei 4 kByte-ROMs beinhaltet, belegt es 8 kByte vom adressierbaren Speicherbereich.

Da die Adress- und Datenleitungen der einzelnen Speicherbausteine (sowohl RAM als auch ROM) parallelgeschaltet sind, muß der jeweilige Baustein mit Hilfe von Auswahlleitungen aktiviert werden - andernfalls wrden alle RAM- und ROM-Bausteine ihre Daten auf den Datenbus legen, was 1. zu einem undefiniertem Signal und 2. zur Überlasung der Bustreiber führen würde.
Die Auswahl kann mit Hilfe einer einzigen Leitung oder durch eine bestimmte Bit-Kombination mehrerer Leitungen erfolgen. Die beiden ATARI-BASIC-ROMs besitzen z.B. jeweils zwei Auswahlleitungen, die mit CS bezeichnet werden. Mit zwei Leitungen lassen sich 2 = 4 verschiedene Bausteine auswählen - die Dekodierung erfolgt intern in den ROMs; daher sind alle anderen Leitungen der ROMs ausnahmslos parallelgeschaltet. Da die EPROMs nur über eine einzige Auswahlleitung verfügen, muß man für eine externe Dekodierung sorgen. Zu diesem Zweck bekommen Sie bei dem Händler, bei dem Sie Ihr EPROM-Programmiergerät gekauft haben, spezielle EPROM-Steckmodule mit einem entsprechenden Dekodierbaustein.

ROMS ODER EPROMS?

Die ROM-Bausteine Ihres ATARI-Systems ließen sich auch durch kompatible EPROMs ersetzen. Warum hat ATARI ROMs anstatt EPROMs verwendet? Dafür gibt es zwei Gründe:

1. In größeren Stückzahlen (schon ab ca. 1000 Stck.) sind ROMs billiger als EPROMs. Ihr Inhalt wird bereits bei der Herstellung als Maske fest einprogrammiert.

2. Bei der Mehrzahl der Anwender dürfte wohl kaum der Wunsch bestehen, das Bertriebssytem, BASIC oder irgendwelche Spiele zu löschen, um sie neu zu programmieren, z.B. in modifizierter Form. ABER GENAU DIES IST DIE CHARAKTRERISTISCHE EIGENSCHAFT DER EPROMS: ihr Inhalt läßt sich mit Hilfe von speziellem UV-Licht LÖSCHEN (EPROM heißt: Erasable Programmable ROM). Anschließend kann man sie mit anderen Daten beschreiben. Da dies auf eine vollkommen andere Weise geschieht als z.B. bei den RAM-Bausteinen, benötigt man dazu ein spezielles Programmiergerät. Mittlerweile gibt es solche Geräte ('EPROM-Burner') bereits auch speziell für die ATARI-Computer; der Anschluß erfolgt meist über die Steuerknüppel-Buchsen.

WIE WIRD EIN EPROM PROGRAMMIERT?

Es sei vorausgeschickt, daß sich die folgenden Betrachtungen auf die EPROMs TMS2516/32 von Texas Instruments beziehen; grundsätzlich gelten sie jedoch auch für Fabrikate anderer Hersteller.

Wenn Sie ein EPROM kaufen, ist es unprogrammiert. Alle Bits befinden sich im Zustand log.1. Schaut man sich mit dem EPROM-Programmiergerät ein leeres EPROM an, findet man daher in allen Worten den Hexadezimalwert 'FF'. Aus diesem Grunde macht der LEERTEST (L) der Betriebssoftware nichts anderes, als alle Speicherzellen daraufhin zu überprüfen, ob sie den Wert 'FF' enthalten.

Für jedes Bit enthält ein EPROM eine Feldeffekt-Transistor-Zelle, deren Gate-Anschluß 'in der Luft hängt' (floating). Im Normalen (unprogrammierten Zustand wird damit eine log.1 repräsentiert. Bringt man mit Hilfe der Programmierung eine Ladung auf das Gate einer Bit-Zelle, schaltet der Transistor durch; es wird eine log.Ø dargestellt. Um ein EPROM mit der gewünschten Information zu versehen, müssen also die entsprechenden Bits der angesprochenen Bytes auf logØ programmiert werden. Dies geschieht folgendermaßn:

1. Im EPROM, daß sich auf dem Stecksockel des EPROM-Programmiergerätes befindet, wird ein Wort adressiert.

2. Die Chip-Auswahlleitung CS (nur 2616) wird auf log.Ø gesetzt, die Programmierspannung Vpp (25 Volt) wird durchgeschaltet.

3. Die zu programmierenden Daten werden über die Schnittstelle parallel auf die Datenleitungen des EPROMs gelegt.

4. Nach Stabilisierung der Daten- und Adressensignale wird ein 50 Millisekunden langer TTL-Impuls (log.1 für 2516, log.0 für 2532) zum Programmieranschluß PGM gesendet. Alle Bits, die log.1 beibehalten sollen, bleiben unverändert; alle übrigen Bits werden durch diesen Vorgang auf log.0 programmiert.

Um den gesamten Detailablauf der Programmierung brauchen Sie sich in der Regel nicht zu kümmern, da dies die Steuersoftware Ihres EPROM-Programmiergerätes für Sie übernimmt. Auch hardwaremäßig werden Sie weitgehend entlastet: die Programmierspannung wird meist mit Hilfe eines DC/DC-WAndlers erzeugt.

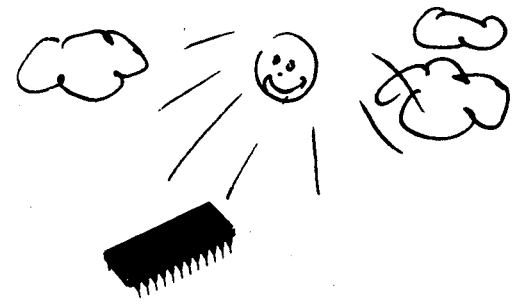
Die Schritte 1 bis 4 lassen sich für sämtliche Adressen des EPROMs durchführen. Bei der Programmierung von Blöcken bzw. kompletten EPROMs wird die Adresse durch einen Adress-Zähler automatisch erhöht. Nach vollendeter Programmierung nehmen die meisten EROM-Burner automatisch einen Vergleichstest zwischen der Vorlage und dem frisch programmierten EPROM vor (Verify).

An anderer Stelle in diesem Buch finden Sie eine Beschreibung des EPROM-Programmiergerätes PROGRAMMER 400/800 und der Möglichkeiten, die sein Betriebssystem bietet.

WIE LÖSCHT MAN EIN PROGRAMMIERTES EPROM?

Die bei manchen Programmiergeräten mögliche Einzelbyte-Programmierung kann auch dazu benutzt werden, ein bereits programmiertes Byte zu korrigieren. Wie bereits erläutert, können jedoch nur Einsen auf log.0 programmiert werden, jedoch nicht umgekehrt. Einsen lassen sich nur durch Bestrahlung mit UV-Licht erzeugen. Dies läßt sich natürlich nicht bitweise, sondern nur für ein KOMPLETTES EPROM durchführen.
Wie bereits erwähnt liegt der Vorteil von EPROMs in der Wiederverwendbarkeit. Durch Bestrahlung des Quarzfensters mit ultraviolettem Licht fließt die Ladung der auf log.0 programmierten Transistorzellen wieder ab, sodaß alle Bits wieder den Wert log.1 annehmen (alle Worte = hex FF). Hierzu ist ein ozonfreier Strahler mit einer bestimmten Wellenlänge (ca. 253 nm) erforderlich. 'Heimsonnen' eignen sich nicht !!! Die minimale Strahlendosis ergibt sich aus (UV-Strahlungsdichte x Löschzeit) und beträgt laut Texas Instruments 15 Wattsekunden/cm . Mit einer typischen filterlosen UV-Lampe löscht man ein EPROM daher in ca. 21 Minuten:

$$t = \frac{15000 \frac{mWs}{cm}}{12 \frac{mW}{cm}} \times \frac{1}{60} = \underline{\underline{20{,}8 \text{ Min.}}}$$



Für Löschlampen mit anderer Strahlungsdichte sowie andere EPROM-Fabrikate ergeben sich entsprechend andere Zeitwerte. Angaben dazu finden Sie in den Datenblättern der EPROM- bzw. Löschlampenhersteller. Im Zweifelsfalle ist es sinnvoller, lieber etwas länger als zu kurz zu belichten - andernfalls kann es zu Programmierungsfehlern oder erst später auftretenden Bit-Fehlern kommen. Grundsätzlich gilt: je stärker die Leistung der Löschlampe, desto kürzer die Löschzeit. Es ist jedoch zu beachten, daß eine zu starke Strahlung das EPROM aufgrund der Hitzeentwicklung schädigen kann. Beim (nicht zu empfehlenden) Selbstbau sollte weiterhin darauf geachtet werden, daß der Abstand zwischen EPROM und Lampe ca. zwei bis drei cm beträgt.

ACHTUNG:

Haut und Augen sind vor der UV-Strahlung der Löschlampe zu schützen. Vor Öffnen das Löschgerät ausschalten bzw. ein Modell mit automatischer Abschaltung bei Öffnung des Deckels benutzen!

EIN PAAR TIPS ZUM UMGANG MIT EPROMS

1. Bitte denken Sie daran, daß EPROMs als MOS-Bauelemente empfindlich gegen elektrostatische Aufladung sind. Vermeiden Sie es möglichst, die Anschlüsse der EPROMs mit den Fingern zu berühren, da auch der menschliche Körper stets einer mehr oder weniger großen statischen Aufladung unterliegt. Dies hängt weitgehend von Umgebungseinflüssen ab (Luftfeuchte, Teppich- und Schuhsohlenmaterial, Kleidung etc.)

Unbenutzte EPROMs vewahrt man entweder in einer antistatischen Röhre oder steckt sie auf antistatischen Schaumstoff (nicht Styropor!) auf. Ist beides nicht zur Hand, hilft eine Blechschachtel oder Alu-Folie.

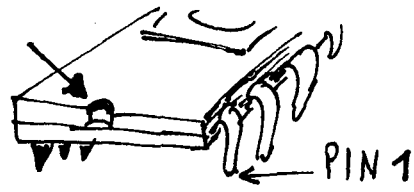
2. Kleben sie die Quarzfenster Ihrer programmierten EPROMs mit einem lichtundurchlässigen Aufkleber (Papierwarengeschäft) zu. während ein auf diese Weise geschütztes EPROM seine Daten über viele Jahre hinaus behält, kann direkte Sonnenstrahlung bereits nach einer Woche zu Datenverlust führen. Um Verwechslungen zu vermeiden ist es sinnvoll, den Aufkleber mit einer Programmbezeichnung zu kennzeichnen.

3. Die Zuverlässigkeit läßt sich erheblich steigern, wenn man alle EPROMs auch vor der Erstprogrammierung löscht.

4. Lassen Sie Ihre EPROMs nach der UV-Löschung erst abkühlen, bevor Sie mit der Neuprogrammierung beginnen.

5. Sorgen Sie beim Betrieb der EPROMs für ausreichende Kühlung, da bei niedriger Betriebstemperatur der Leckstrom klein bleibt. Auch die offene Bauweise der lieferbaren EPROM-Steckmodule kommt dieser Forderung entgegen.

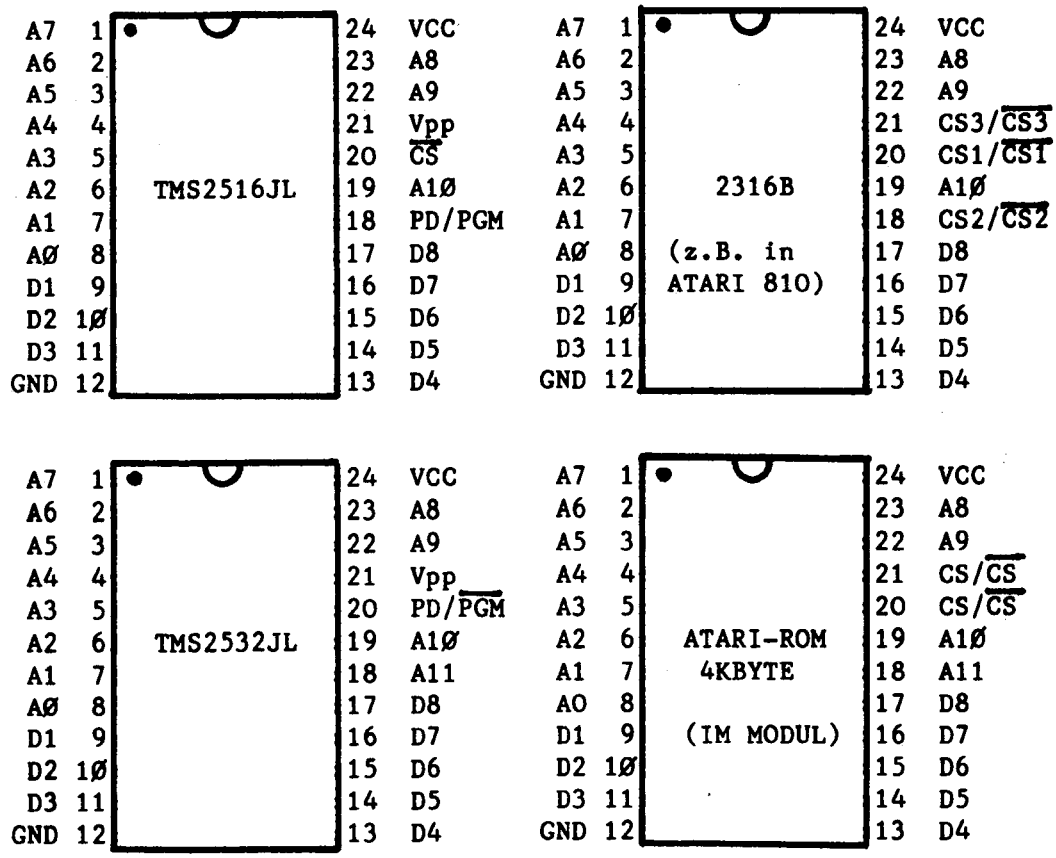
6. Achten Sie beim Einstecken der EPROMs in den Sockel des EPROM-Programmiergerätes bzw. in die Leer-Platinen darauf, daß dies richtig herum geschieht. Bei vielen EPROMs ist Pin 1 gekennzeichnet. Andernfalls richten Sie sich nach der Kerbe an einer der Schmalseiten:



23

7. Programmieren Sie mit Ihrem EPROM-Programmiergerät nur solche EPROMs, für die es entwickelt wurde.

ANSCHLUßBELEGUNG VON ROMS UND EPROMS



Erklärung der Abkürzungen:

A(N): ADRESSEN-EINGÄNGE
D(N): DATEN-AUSGÄNGE,BEI EPROMS AUCH EINGÄNGE BEIM PROGRAMMIEREN
PD/PGM: POWER DOWN/PROGRAM
Vcc: VERSORGUNGSSPANNUNG (+5 VOLT)
Vpp: PROGRAMMIERSPANNUNG (+25 VOLT)
GND: GROUND (MASSE)
CS: CHIP-SELECT (AUSWAHL), AKTIV LOG.1
CS: CHIP-SELECT (AUSWAHL), AKTIV LOG.0
CSx/CSx: CHIP-SELECT, PROGRAMMIERBAR (d.h. bei der Herstellung des ROMs wird festgelegt, ob der Anschluß aktiv 0 oder 1 ist)

B VERGLEICHSTYPEN (Auszugsweise, o. Gewähr)

1. FÜR TMS2516JL:

HN462716	HITACHI
2716	INTEL
MKB2716	MOSTEK
MBM2716	FUJITSU
uPD2716	NEC
SY2716	SYNERTEC

2. FÜR TMS2532JL:

TMS2532	INTEL
MCM2532	MOTOROLA
NMC2532	NATIONAL SEMICONDUCTOR
HN462532	HITACHI
MSM2532	OKI

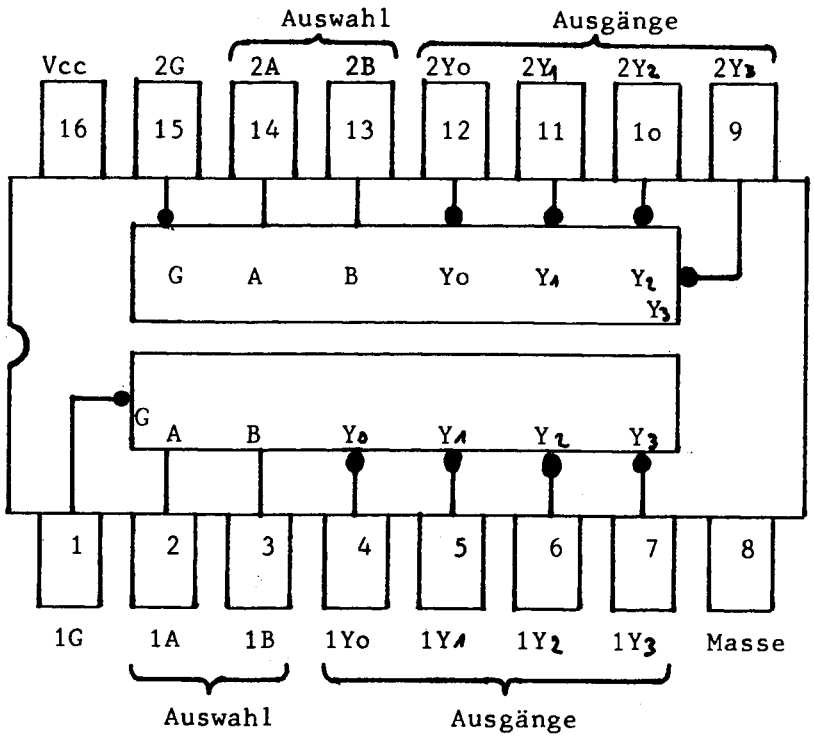
UMBAU EINES ATARI-ROMMODULS ZUR AUFNAHME VON EPROMS

Wer für seinen ATARI über ein EPROM-Programmiergerät verfügt, möchte die 'selbstgebrannten' Speicherbausteine natürlich auch in seinem Computer arbeiten lassen. Sollten keine speziellen EPROM-Leermodule aufzutreiben sein, bietet es sich an, zu diesem Zweck ein ATARI-ROM-Modul zu 'schlachten'. Bestimmt findet sich dazu irgendein Spiel, das nach hundertmaligem Gebrauch etwas langweilig geworden ist.

Also: ATARI-ROMs raus, EPROMs rein. Allerdings muß auf der Platine zuvor noch eine kleine Änderung vorgenommen werden. Wie an anderer Stelle bereits erwähnt, verfügen die von ATARI verwendeten ROMs über interne Chip-Auswahlleitungen und die EPROMs nicht. Daher müssen wir dafür sorgen, daß extern erkannt wird, welcher der beiden Bausteine angesprochen werden soll. (Will man nur ein einziges EPROM 2532 einsetzen, ist keinerlei Modifikation notwendig - auch ist es egal, in welchen der beiden Sockel es gesteckt wird).

Somit besteht die Änderung darin, daß wir die zu den Sockeln führenden Auswahlleitungen CS1 und CS3 an geeigneter Stelle unterbrechen und einen Dekoder zwischenschalten. Dazu eignet sich z.B. der 2-Bit-Binärdekoder 74LS139 (siehe Abb 24).

Anschlußbelegung des 74LS139 (Texas Instruments u.a.):



G = Freigabesignal

24

Wegen des geringen Preises soll es uns nicht weiter stören, daß wir nur die eine Hälfte des Bausteins benutzen. Die beiden Leitungen, auf die es ankommt, führen zu den ROM-Anschlüssen 20 und 21 (siehe Abb.25):

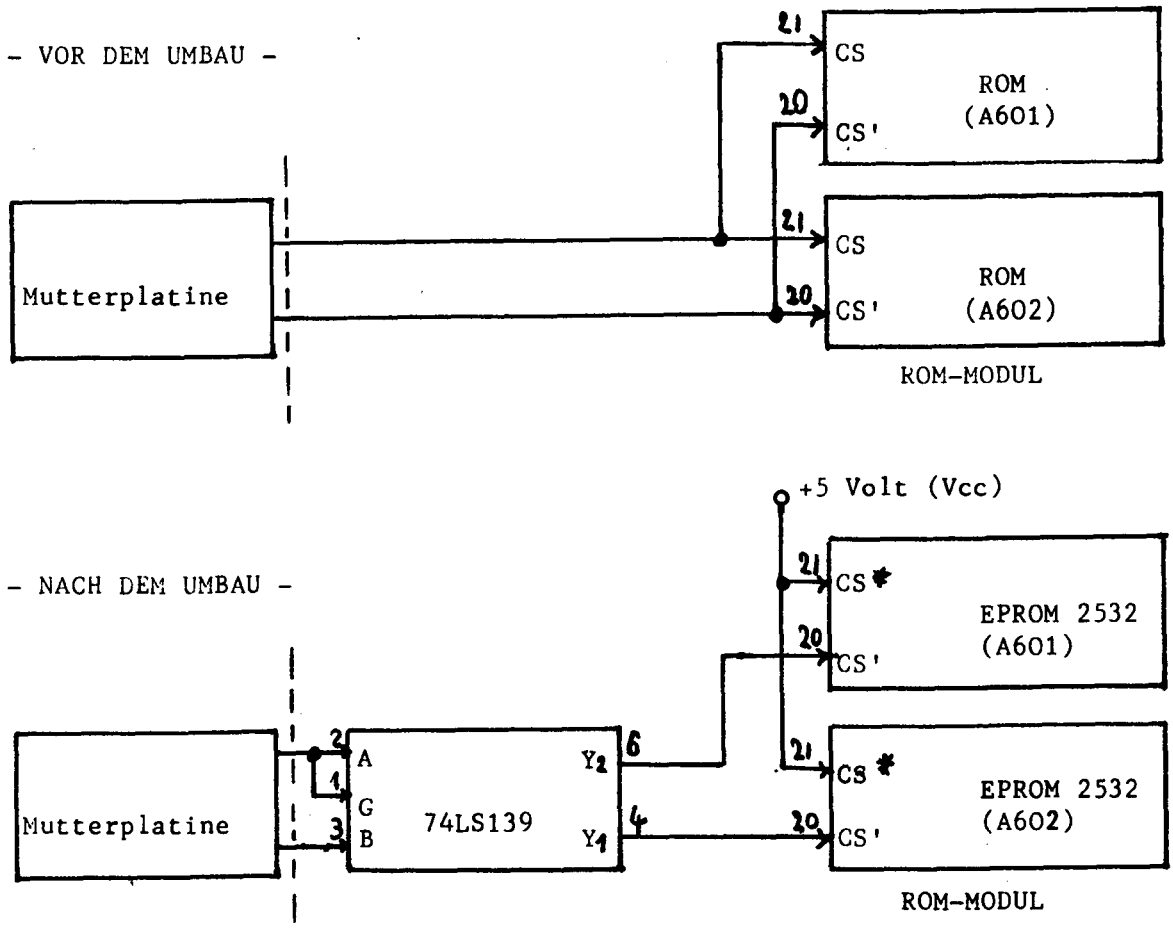


Abbildung 25 Blockschaltbild des Modul-Umbaus

Das EPROM 2532 verfügt über nur eine Chipauswahlleitung; der Dekoder sorgt dafür, daß jeweils nur eines der beiden EPROMs angewählt werden kann, je nach Pegel der beiden Auswahlleitungen von der Mutterplatine. Der mit * gekennzeichnete Anschluß 21 ist beim 2532 die PD/PGM-Leitung; sie muß zum Lesen der Bausteine auf log.1 liegen und wird daher fest mit +5 Volt verbunden.

Alles, was Sie zur Umrüstung Ihres ROM-Moduls brauchen: Schraubenzieher, Seitenschneider, scharfes Messer, Lötkolben und etwas Lötzinn. Oder einen Freund der dies alles hat. Wenn Sie sich genau an die nun folgenden Schritte halten, kann eigentlich garnichts schiefgehen.

Beginnen wir also mit dem Umbau des ROM-Moduls:

1. Schrauben Sie das Modul-Gehäuse auf und nehmen Sie die Platine heraus (Achtung, nichts zerbrechen!).

2. Hebeln Sie die beiden ROMs mit dem Schraubenzieher aus ihren Fassungen. Sollte es sich um ein Spiel mit nur einem ROM handeln, löten Sie einen zweiten Sockel ein.

3. Änderungen auf der Vorderseite (Bestückungsseite, Abb.26):

3.1. Trennen Sie mit dem Messer bei a vorsichtig die Leiterbahn auf, die Pin 20 von A601 mit Pin 20 von A602 verbindet.

3.2. Verbinden Sie Pin 21 von A602 mit Vcc, indem Sie mit dem Messer auf der äußeren breiten Leiterbahn bei b etwas Lötstoplack entfernen und von dort eine Lötbrücke zur Lötstelle von Pin 21 fließen lassen.

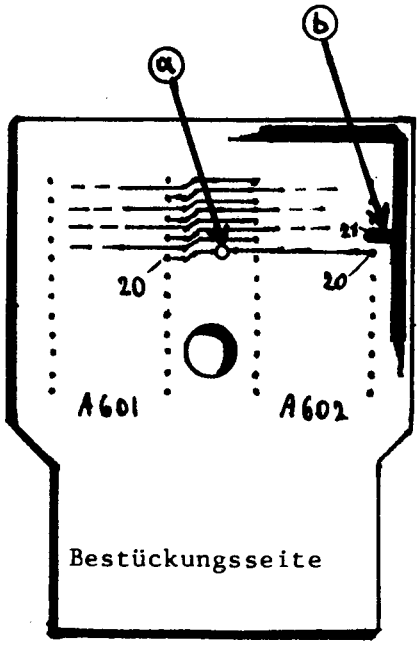


Abbildung 26

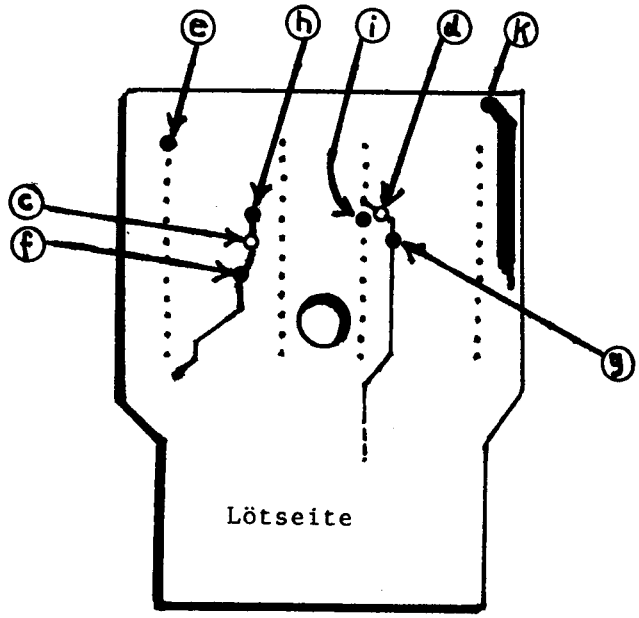


Abbildung 24

4. Änderungen auf der Rückseite (Lötseite, Abb.27):

4.1 Unterbrechen Sie die gezeigten Leiterbahnen bei c und d.

4.2. Bereiten Sie e bis k als Lötstellen vor, indem Sie etwas frisches Lötzinn auftragen (ggf. vorher mit dem Messer Lötstoplack entfernen).

5. Schneiden Sie sich nun sechs ca. 60 mm lange Stücke dünner isolierter Litze zurecht. Löten Sie je eines an die in Abb. 28 gezeigten Anschlüsse des 74LS139. Die Pins 1 und 2 müssen verbunden werden.

6. Auf der ROM-Platine ist zwischen den beiden 24-poligen Sockeln genug Platz, daß Sie dort das IC mit der beschrifteten Seite nach unten auf die Platine kleben können. Ein Tropfen Alleskleber genügt. Die Kerbe des ICs soll dabei zur oberen Kante der Platine zeigen. Die genaue Positionierung geht aus Abb. 29 hervor.

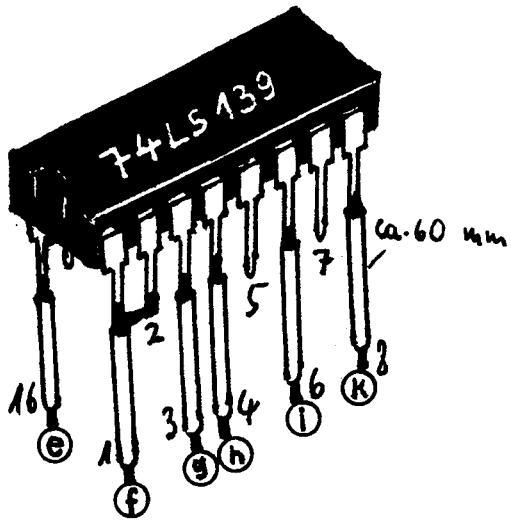


Abbildung 28

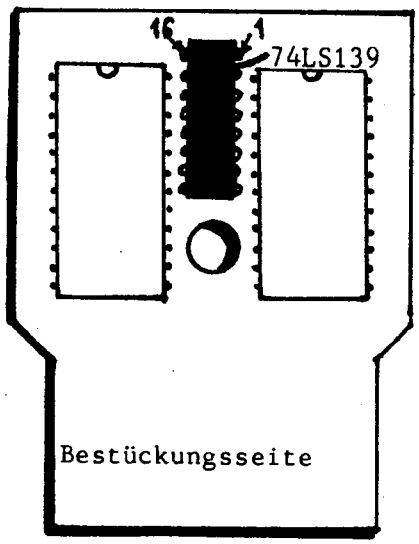


Abbildung 29

7. Nun müssen Sie noch die Litzen e bis k mit den gleichlautend gekennzeichneten Lötpunkten auf der Platinen-Rückseite verbinden, also e mit e, f mit f usw. Dazu werden die Litzen zuerst nach Augenmaß auf die jeweils passende Länge gekürzt, am Ende ca. 2 mm abisoliert und mit Lötzinn benetzt. Die Drähte lassen sich dann ohne langes 'Schmoren' mit der Platine verlöten.

8. Nachdem Sie alle Verbindungen (und Unterbrechungen) nochmals sorgfältig überprüft haben, können Sie Ihre EPROMs einsetzen und die Platine testen. DAS EPROM MIT DEM HÖHEREN ADRESSBEREICH KOMMT IN DEN SOCKEL A601. Bei Betrieb ohne Gehäuse ist unbedingt darauf zu achten, daß die Platine mit der Lötseite nach vorne (zur Tastatur zeigend) in den Modulschacht des Computers gesteckt wird.

9. Zum Schutz vor mechanischen Beschädigungen kann die Platine wieder in ihrem Gehäuse montiert werden. Man sollte darauf achten, dabei keine Drähte abgerissen werden. Sicherheitshalber sollte man auch etwas Isolierband auf die Innenseite des Modul-Deckels kleben, um Kontakt mit den Anschlüssen des 74LS139 zu vermeiden.

ZAHLENUMWANDLUNG TEIL1: DIE TABELLE

④		③		②		①	
dez.	hex.	dez.	hex.	dez.	hex.	dez.	hex
4096	1	256	1	16	1	1	1
8192	2	512	2	32	2	2	2
12288	3	768	3	48	3	3	3
16384	4	1024	4	64	4	4	4
20480	5	1280	5	80	5	5	5
24576	6	1536	6	96	6	6	6
28672	7	1792	7	112	7	7	7
32768	8	2048	8	128	8	8	8
36864	9	2304	9	144	9	9	9
40960	A	2560	A	160	A	10	A
45056	B	2816	B	176	B	11	B
49152	C	3072	C	192	C	12	C
53248	D	3328	D	208	D	13	D
57344	E	3584	E	224	E	14	E
61440	F	3840	F	240	F	15	F

Umwandlungstabelle

Beispiele:

*** Umwandlung hexadez. -- dezimal ***

```
                ④③②①
$E45F  =  E 4 5 F
                  |  |  |  |
                  |  |  |  |  15
                  |  |  |  80
                  |  |  1024
                  |  57344
                  Summe = 58463  dez.
```

```
                ②①
$A9 = A 9
                |  |
                |  |  9
                |  160
                  169  dez.
```

*** Umwandlung dezimal -- hexadez. ***

```
 58493  = hex.?
-57344 (nächstkleinerer Wert in ④) → E
= 1149
- 1014 (nächstkleinerer Wert in ③) → 4
=  125
-  112 (nächstkleinerer Wert in ②) → 7
=   13
       (Rest steht in ①) ———————→ D
```

→ $E47D hexadez.

ZAHLENUMWANDLUNG TEIL2: PROGRAMMBEISPIELE IN BASIC

Einige Zahlenwandlungs-Anwendungen in BASIC:

```
10 REM DARSTELLUNG DER ZWEI BCD-WERTE EINES BYTES
20 A=INT(PEEK(1536)/16):REM BCD HIGH
30 B=PEEK(1536)-16*A:REM BCD LOW
```

****

```
10 REM BCD - DARSTELLUNG EINER ZWEISTELLIGEN
20 DEZIMALZAHL (Z.B. DATUM) IN ZWEI BYTES
30 POKE 1537,INT(DD/10):POKE 1536,DD-(INT(DD/10)*10)
```

****

```
10 REM DEZIMAL-DARSTELLUNG EINES ZWEI-BYTE-WERTES (Z.B. ADRESSE)
20 DEZ=PEEK(LO)+256*PEEK(HI)
```

****

```
10 REM UMWANDLUNG EINER DEZIMALZAHL IN EINE ZWEI-BYTE-ADRESSE
15 REM Z.B. DEZ=1536
20 HI=INT(DEZ/256):LO=DEZ-HI*256
```

****

```
10 REM RUNDUNG AUF 2 NACHKOMMA-STELLEN (DM UND PFG.)
20 A=INT((A+5.0E-03)*100)/100
```

****

```
10 REM DIE UMRECHNUNG IN DEZIMALZAHLEN SPAREN:
20 REM ROUTINE ZUM LADEN VON HEX-DATEN
30 REM FUER MASCHINENPROGRAMME U.A.
40 DIM HEX$(3)
50 MLPROG=1536:REM ANFANGSADRESSE Z.B. $0600
60 GRAPHICS 0:POKE 752,1
70 ? "ES WIRD MASCHINEN-CODE GELADEN":?
80 FOR ADDR=MLPROG TO MLPROG+13:REM HIER ANZAHL DER HEX-BYTES ANGEBEN
90 REM AKUSTISCHE ANZEIGE DES LADEVORGANGS
100 READ HEX$:SOUND 0,ADDR-MLPROG,10,10
110 HI=ASC(HEX$(1,1))-48
120 IF HI>9 THEN HI=HI-7
130 LO=ASC(HEX$(2,2))-48
140 IF LO>9 THEN LO=LO-7
150 POKE ADDR,HI*16+LO:NEXT ADDR
160 ? "FERTIG GELADEN.":? :POKE 752,0
170 ? "START DES MASCHINENPROGRAMMS MIT"
180 ? :? "X=USR(";MLPROG;")"
190 REM HIER DIE DATEN DIREKT IN HEX ANGEBEN
200 DATA 68,A9,FF,4C,00,06,60,A5,14
210 DATA 85,15,20,A0,06
```

'ZU FUß' DISASSEMBLIEREN MIT DER OPERATIONSCODE-TABELLE

Beim Assemblieren eines Programms werden bekanntlich die leicht zu merkenden (mnemonischen) Kurzbefehle in Operations-Codes umgewandelt, die von der CPU verstanden werden. Je nach Adressierungsart setzt sich ein Befehl aus ein bis drei solcher Codes zusammen. Hier ein paar Beispiele:

Assembler-Code	Operationscodes	
ROR A	6A	Ein-Byte-Befehl
LDA FF	A9 FF	Zwei-Byte-Befehl
JMP 06A0	4C A0 06	Drei-Byte-Befehl

Ein Disassembler tut genau das Gegenteil: er setzt die Binär-Codes, die selbst in hexadezimaler Schreibweise kaum verständlich sind, wieder in verständliche Assembler-Befehle um. Auch die Art der Adressierung (absolut, indirekt etc.) geht aus der zurückgewonnenen Assembler-Darstellung hervor; so gibt der rückübersetzte Befehl STA (FE),Y z.B. indirekt indizierte Adressierung unter Verwendung des Y-Registers an. Näheres dazu finden Sie in den zahlreichen Handbüchern, die sich mit der 6502-Programmierung in Maschinensprache beschäftigen (siehe Literaturhinweise).

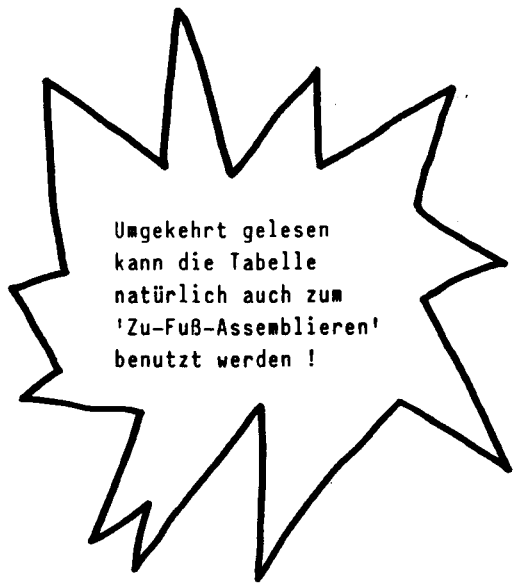
Für die Untersuchung nur weniger Bytes oder oder eines gedruckten Hexdump-Listings ist ein Disassembler uninteressant. Hier hilft uns die 65o2-OPERATIONSCODE-TABELLE auf der folgenden Seite. Sozusagen ein 'Zu-Fuß-Disassembler'. Das Lesen der Tabelle ist einfach: man zerlege das (hexadezimale) Byte in sein nieder- und höherwertiges Halb-Byte und schaue nach, welcher Befehl sich im Schnittpunkt der beiden Koordinaten findet.

Beispiel: Für das Byte A9 finden wir

JMP	← Assemmbler-Befehl
abs	← Adressierungsart

Die Erklärung der in der Tabelle verwendeten Abkürzungen geht aus der untenstehenden Übersicht hervor. Das Operanden-Format zeigt uns auch an, wieviele Bytes zum Befehl noch hinzukommen (im Falle unseres Beispiels '4C' gibt $bbaa an, daß aufgrund der absoluten Adressierung noch zwei Bytes folgen).

Kürzel	Adressierungsart	Operanden-Format
imm	Immediate	#$aa
abs	Absolut	$aabb
zer	Zero-Page	$aa
imp	Impliziert	
ind	Indirekt-absolut	($aabb)
abx	Absolut-indiziert,X	$aabb,X
aby	Absolut-indiziert,Y	$aabb,Y
zpx	Zero-Page-indiziert,X	$aa,X
zpy	Zero-Page-indiziert,Y	$aa,Y
i,x	Indiziert-indirekt,X	($aa,X)
i,y	Indirekt-indiziert,Y	($aa,Y)
rel	Relativ	$aa
akk	Akkumulator	A
.-	Nicht als Operationscode verwendet	



6502 - OPERATIONSCODE - TABELLE

Niederwertiges Halb-Byte ↓ — Höherwertiges Halb-Byte →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK imp	BPL rel	JSR abs	BMI rel	RTI imp	BVC rel	RTS imp	BVS rel	-	BCC rel	LDY imm	BCS rel	CPY imm	BNE rel	CPX imm	BEQ rel
1	ORA i,x	ORA i,y	AND i,x	AND i,y	EOR i,x	EOR i,y	ADC i,x	ADC i,y	STA i,x	STA i,y	LDA i,x	LDA i,y	CMP i,x	CMP i,y	SBC i,x	SBC i,y
2	-	-	-	-	-	-	-	-	-	-	LDX imm	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	BIT zer	-	-	-	-	-	STY zer	STY zpx	LDY zer	LDY zpx	CPY zer	-	CPX zer	-
5	ORA zer	ORA zpx	AND zer	AND zpx	EOR zer	EOR zpx	ADC zer	ADC zpx	STA zer	STA zpx	LDA zer	LDA zpx	CMP zer	CMP zpx	SBC zer	SBC zpx
6	ASL zer	ASL zpx	ROL zer	ROL zpx	LSR zer	LSR zpx	ROR zer	ROR zpx	STX zer	STX zpy	LDX zer	LDX zpy	DEC zer	DEC zpx	INC zer	INC zpx
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	PHP imp	CLC imp	PLP imp	SEC imp	PHA imp	CLI imp	PLA imp	SEI imp	DEY imp	TYA imp	TAY imp	CLV imp	INY imp	CLD imp	INX imp	SED imp
9	ORA imm	ORA aby	AND imm	AND aby	EOR imm	EOR aby	ADC imm	ADC aby	-	STA aby	LDA imm	LDA aby	CMP imm	CMP aby	SBC imm	SBC aby
A	ASL akk	-	ROL akk	-	LSR akk	-	ROR akk	-	TXA imp	TXS imp	TAX imp	TSX imp	DEX imp	-	NOP imp	-
B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
C	-	-	BIT abs	-	JMP abs	-	JMP ind	-	STY abs	-	LDY abs	LDY abx	CPY abs	-	CPX abs	-
D	ORA abs	ORA abx	AND abs	AND abx	EOR abs	EOR abx	ADC abs	ADC abx	STA abs	STA abx	LDA abs	LDA abx	CMP abs	CMP abx	SBC abs	SBC abx
E	ASL abs	ASL abx	ROL abs	ROL abx	LSR abs	LSR abx	ROR abs	ROR abx	STX abs	-	LDX abs	LDX aby	DEC abs	DEC abx	INC abs	INC abx
F	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

BASIC-BEFEHLE

BEFEHL	PARAMETER	FUNKTION	BEMERKUNGEN
>A	(Zeilennr.) (Schrittweite)	Automat. Zeilennumerierung	Abbruch mit 'Break'
>D	(Zeilennr.VON) (Zeilennr.BIS)	Löschen von Zeilenblöcken	
>M	(linker Rand) (rechter Rand)	Bildschirmrand-Änderung	Nur von 0 bis 39!
>E		Umbelegung der Cursor-Tasten	Zurück: nochmals >E
>#	(Dezimalzahl)	Umwandlung Dezimal -- Hex	Max. 65535
>$	(Hex-Zahl)	Umwandlung Hex -- Dezimal	Max. FFFF
>T	(Startadr.) (Endadr.)	RAM-Test	Angaben in Hex
>R	(Startzeilennr.) (Schrittweite)	Zeilen-Renumerierung	Max. 478 Zeilen
>*		Sprung zum Monitor	Maschinensprache

MONITOR-BEFEHLE (MASCHINENSPRACHE)

Alle Angaben in HEX

BEFEHL	PARAMETER	FUNKTION	BEMERKUNGEN
.		Befehlseingabe-Zeichen	
M	XXXX YYYY	Speicherinhalt anzeigen	Start-/Endadresse
R		Register anzeigen	CPU-Register (6502)
:	--	Speicherzellen ändern	m. Cursor überschreiben
;		6502-Register ändern	m. Cursor überschreiben
G	XXXX	GOTO (Programm-Start)	Mon.-Rücksprung auf BRK
S	XXXX YYYY	Auf Cassette abspeichern	Endadr.+1 angeben !
L	XXXX	Von Cassette laden	Ab beliebiger RAM-Adr.
H	XXXX YYYY 'ZZZZZ...	String-Suche (ASCII)	Max. 20 Zeichen
H	XXXX YYYY ZZ ZZ ZZ ..	Code-Suche (Hex)	Max. 20 Bytes
D	XXXX	Disassemblieren	Abbrechen m. Leertaste
,	--	Disass.-Listing ändern	Hex-Code m. Cursor überschreiben (max. 3 Bytes)
B		Sprungweiten-Berechnung	Für relative Adressen
A		Rücksprung ins ATARI-DOS	Ohne DOS -- MEMO PAD
X		Rücksprung zum BASIC	
?	--	FEHLER	Falscher Befehl, unkorr. Hex-Daten oder Versuch, ROM zu ändern

Monkey Wrench ist eingetr. Warenzeichen der Eastern House Inc. 3239 Linda Dr.,Winston Salem, N.C.2706/USA Tel. 919-924-2889 oder -748-8446

DOS-TIPS

In diesem Kapitel erfahren Sie einige Tips und Tricks, die Ihnen ein noch effektiveres Arbeiten mit dem ATARI-DOS 2.0 S ermöglichen sollen

● Benutzt man im DOS den Befehl 'C' (Copy File), lautet das Befehlsformat

D1:NAME,D2:NAME.

Völlig ausreichend ist aber die Kurzform

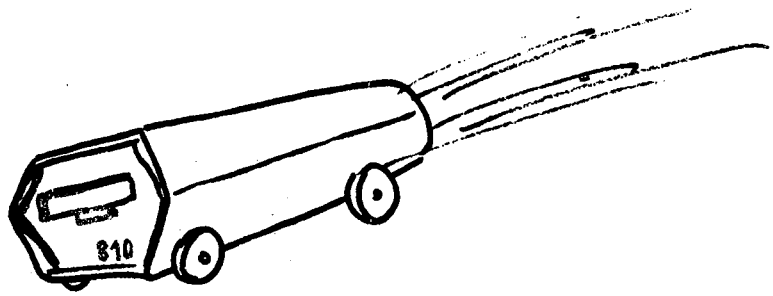
NAME,2:NAME

● Der 'C'-Befehl läßt sich auch dazu verwenden, einen Text per Tastatur 'direkt' auf Diskette zu schreiben. Nach 'C' gibt man dazu ein:

E:,D:TEXTNAME

Eingetippter Text wird nun auf die Diskette geschrieben, wobei die Zeilenlänge maximal 128 Zeichen (Länge des Sektor-Puffers) betragen kann. Abgeschlossen wird die Eingabe mit CONTROL-3 (End Of File).

● Beim Kopieren mit nur einem Laufwerk: Will man alle Dateien bis auf jene mit der Erweiterung '.SYS' auf eine andere Diskette übertragen, gibt man als Dateinamen einfach '*.*' an. - Taucht eine weitere Datei auf, die man nicht mit übernehmen möchte, drückt man die BREAK-Taste, wenn DOS 'Insert Destination Disk' verlangt. Die betroffene Datei wird dann übersprungen.



● Das Schreiben auf die Diskette läßt sich beschleunigen, indem man

POKE 1913,80

eingibt, bevor man mit SAVE bzw. LIST abspeichert. Auch wenn man anschließend ins DOS geht, laufen die mit Schreiboperationen verbundenen Befehle schneller ab (z. B. DUPDISK). Das Geheimnis: mit obigem POKE setzt man die Routine außer Kraft, mit der die Information nach dem Schreiben normalerweise auf korrekte Aufzeichnung überprüft wird (Read After Write Verify). Deshalb empfiehlt sich dieses Verfahren auch nur bei der Verwendung qualitativ hochwertiger Disketten und exakt eingestellten Disk-Laufwerken. Man kann sich die DOS-Änderung übrigens dauerhaft erhalten, indem man anschließend das DOS mit 'H' neu abspeichert - jedoch sollte man auf dem Disketten-Etikett einen entsprechenden Vermerk machen.

NOCH MEHR DOS-TIPS

● Spätestens beim Versuch, modifizierte DOS-Versionen zu kopieren, werden Sie feststellen: Nach dem 'C'-Befehl die Parameter

D1:DOS.SYS,D2:DOS.SYS

anzugeben, funktioniert nicht. Erlaubt ist jedoch

D1:DOS.SYS,D2:DOS.NEU

Nimmt man dann mit 'E' eine Umbenennung, nämlich

D2:DOS.NEU,D2:DOS.SYS ,

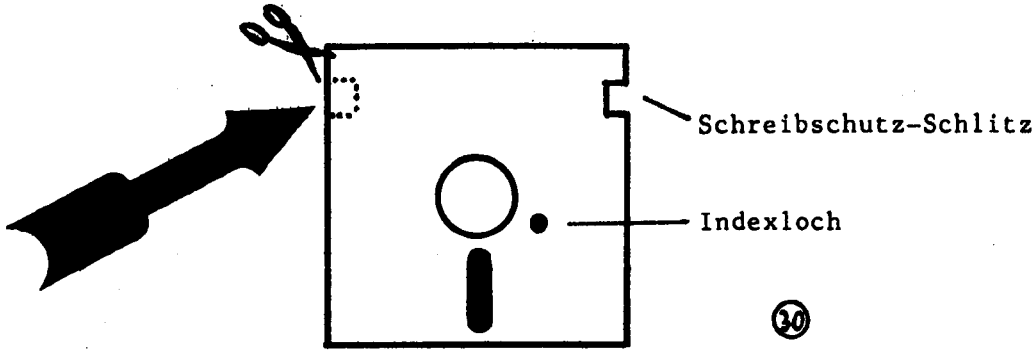
hat man das Gleiche erreicht!

● Das Problem: Sie wollen alle Dateien einer Diskette (bis auf xxxx.SYS) auf eine andere Diskette schreiben, auf der sich bereits einige Dateien befinden, die Sie nicht löschen wollen. Geben Sie nach 'C' (Copy File) ein:

D1: .*,D2:

Die neue Diskette wird dann nicht überschrieben, sondern mit dem Inhalt der alten Diskette ergänzt. Zumindest überschlägig sollte man dabei jedoch stets rechnen, ob auch genug fre ie Sektoren zur Aufnahme aller Dateien zur Verfügung stehen - andernfalls kommt der bedauerliche ERROR 162 (Diskette voll).

● Viele Disketten lassen sich auch auf der Rückseite beschreiben!!! Man muß dazu lediglich - z.B. mit einer Nagelschere - vorsichtig einen Schreibschutz-Schlitz ausschneiden, wie er auf der gegenüberliegenden Seite bereits fabrikmäßig ausgestanzt wurde (siehe Abbildung).



Nicht alle Fabrikate eignen sich für diese Sparmaßnahme. Gute Erfahrungen machten Atari-Anwender bisher mit den Marken Maxell, BASF und Memorex (ohne Anspruch auf Vollständigkeit). Das Verfahren funktioniert übrigens nur bei Verwendung der Diskettenstation ATARI 810, die im Gegensatz zu den meisten anderen Laufwerken - insbesondere solchen mit doppelter Aufzeichnungsdichte - das Indexloch nicht abfragt.

STECKVERBINDER FÜR ATARI: TYPENBEZEICHNUNGEN

Wer beispielsweise schon mal nach passenden Steckern für die Steuerknüppel-Anschlüsse gesucht hat, kennt das Problem: ohne genaue Typenbezeichnung geht nichts. Schon garnicht, wenn man telefonisch nach einer Bezugsquelle forscht. Schluß damit. Hier finden Sie die passenden Typen; einige werden auch von anderen Herstellern angeboten, allerdings mit anderen Bezeichnungen.

1. Steckverbinder für ATARI 400/800

a) 9-pol. Buchse für die Steuerknüppel-Anschlüsse
Hersteller: Harting
Bestell-Nummer: 09 67 009 2704
Gehäuse dazu: 09 67 009 0411

b) 15-pol. Buchse für die serielle Schnittstelle:
Hersteller: Molex
Bestell-Nummer: keine. Hier hilft z.Zt. nur Selbermachen (passende Kontakte aus anderen Buchsen auf Lochrasterplatten-Abschnitt einlöten)

c) 5-pol.Stecker für Monitor- und NF-Anschluß:
Hersteller: Hirschmann u.a.
Bezeichnung: 5-poliger Diodenstecker (Stereo)

2. Steckverbinder für ATARI 850 (Interface-Box)

a) 9-pol. Stecker für die seriellen RS232-Ports:
Hersteller: Harting
Bestell-Nummer: 09 67 009 2604
Gehäuse dazu: 09 67 009 0411

b) 15-pol. Stecker für den parallelen Drucker-Anschluß:
Hersteller: Harting
Bestell-Nummer: 09 67 015 2604
Gehäuse dazu: 09 67 015 0411

3. Steckverbinder für Drucker

a) 36-pol. Stecker für Epson, Seikosha etc.:

Zum Einpressen von Flachbandkabel:

Hersteller: 3M (Scotchflex)
Bestell-Nummer: 3366-1001
Flachbandkabel: 3365-36

Für Lötanschluß:

Hersteller: Amphenol
Bestell-Nummer: 57-30360

b) 40-pol. Leiterplattensteckverbinder für ATARI 825 (Centronics 737):
Hersteller: 3M (Scotchflex)
Bestell-Nummer: 3464-0001
Flachbandkabel: 3365-40

STECKVERBINDER FÜR ATARI: ANSCHLUSSBELEGUNGEN DER VERBINDUNGSKABEL

1. ATARI 850 - parallel / EPSON,Seikosha etc.

EPSON	ATARI 850	Signal
1	1	Data Strobe
2	2	Datenbit 0
3	3	Datenbit 1
4	4	Datenbit 2
5	5	Datenbit 3
6	6	Datenbit 4
7	7	Datenbit 5
8	8	Datenbit 6
	9	Data pull-up (+5 V)
	12	Fault
9	15	Datenbit 7
11	13	Busy
14	11	Ground (Masse)
16		

Im Epson-Stecker muß Pin 14 mit Pin 16 verbunden werden.
Im ATARI-Stecker muß Pin 9 mit Pin 12 verbunden werden.

2. ATARI 850 - seriell (PORT 1) / Modem (ATARI 830 o.ä)

Modem	ATARI 850	Signal
1	20	DTR, Out
2	8	CRX, In
3	2	Send Data, Out
4	3	Receive Data ,In
5	7	Signal Ground
6	6	DSR, In
=	4	RTS, Out
8	5	CTS, In

Der Modem-Anschluß erfolgt über Port 1, also an der Interface-Box ganz rechts. Für das Modem wird ein 25-poliger Min-D-Stecker benötigt (z.B. Harting 09-67-025-2604, Gehäuse: 09-67-025-0411). Für ATARI 850 braucht man den zuvor unter 2a beschriebenen Stecker. Desweiteren ist natürlich noch eine entsprechende Treiber-Software erforderlich. Atari's ROM-Modul Telelink (tm) ist für den deutschen Markt ungeeignet, da es auf die amerikanischen EDV-Informationsnetze zugeschnitten ist. Empfehlenswert ist z.B. 'FRIENDLY TERMINAL' von Harry Coons, 1123 Oakfair Lane, Harbor City, CA90710. Dieses Kommunikationsprogramm gestattet sowohl den direkten Dialog als auch die Übertragung von Files. Bei einem 48k-System stehen dem Anwender über ca. 20 k RAM als Pufferspeicher zur Verfügung.

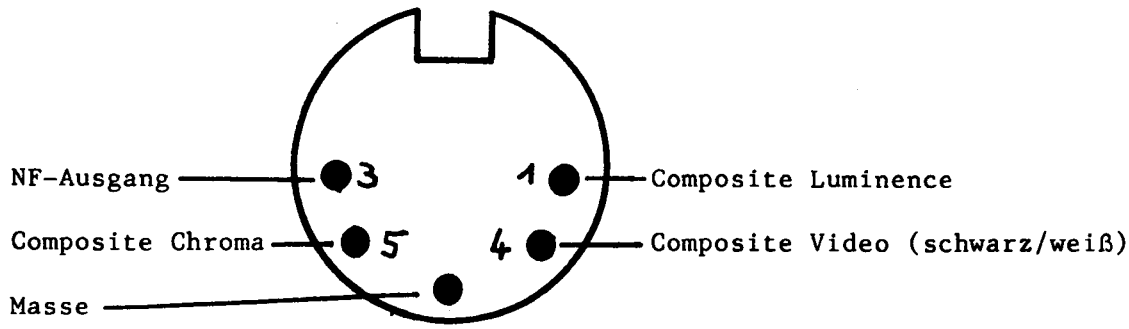
WICHTIG: FÜR DEN BETRIEB EINES MODEMS AM ÖFFENTLICHEN TELEFONNETZ SIND DIE JEWEILS GELTENDEN BESTIMMUNGEN DER DEUTSCHEN BUNDESPOST ZU BEACHTEN.

3. Anschluß des ATARI 800 an Video-Monitor und Stereo-Anlage.

Leider verfügt nur der ATARI 800 über die Möglichkeit, einen preiswerten Monitor (z.B. Zenith ZVM-121-E) anzuschließen. Auch das Erlebnis, Weltraumspiele wie STAR RAIDER (eingetr. Warenzeichen ATARI Inc.) noch realistischer zu gestalten, bleiben dem 400er-Besitzer versagt. Vielleicht bietet eines Tages jemand einen entsprechenden Nachrüstsatz an?

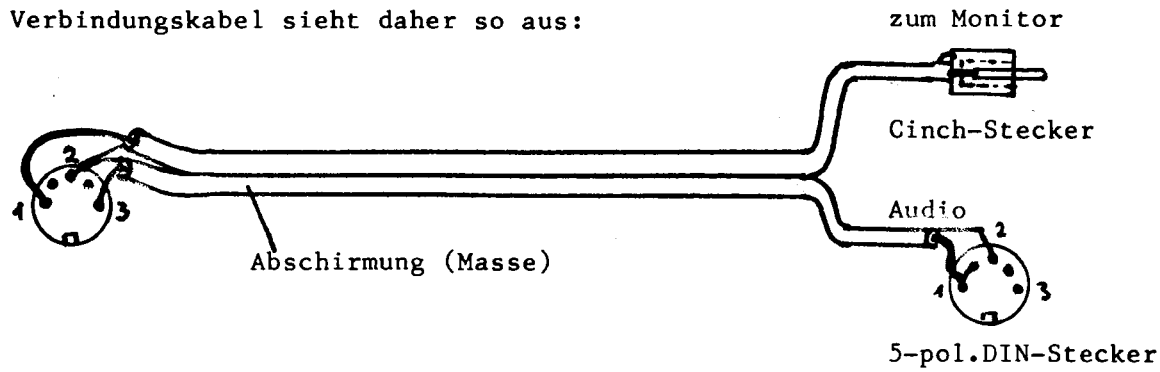
Der Anschluß an den ATARI 800 ist einfach: Sie benötigen lediglich einen 5-Poligen Diodenstecker (Stereo), etwas zweipoliges abgeschirmtes Kabel und die Stecker für Ihren (Stereo-)Verstärker bzw. Monitor. Für den NF-Anschluß ist meist ebenfalls die Verwendung 5-poliger Diodenstecker vorgesehen. Der Zenith-Monitor benötigt, wie viele vergleichbare Geräte, einen sog. Cinch-Stecker. Andere Fabrikate brauchen einen BNC-Stecker. Soll der Monitor erst angeschafft werden, sollte man gleich einen passenden Stecker dazu kaufen.

Die ATARI-800-Buchse ist folgendermaßen beschaltet:



31

Unser Verbindungskabel sieht daher so aus:



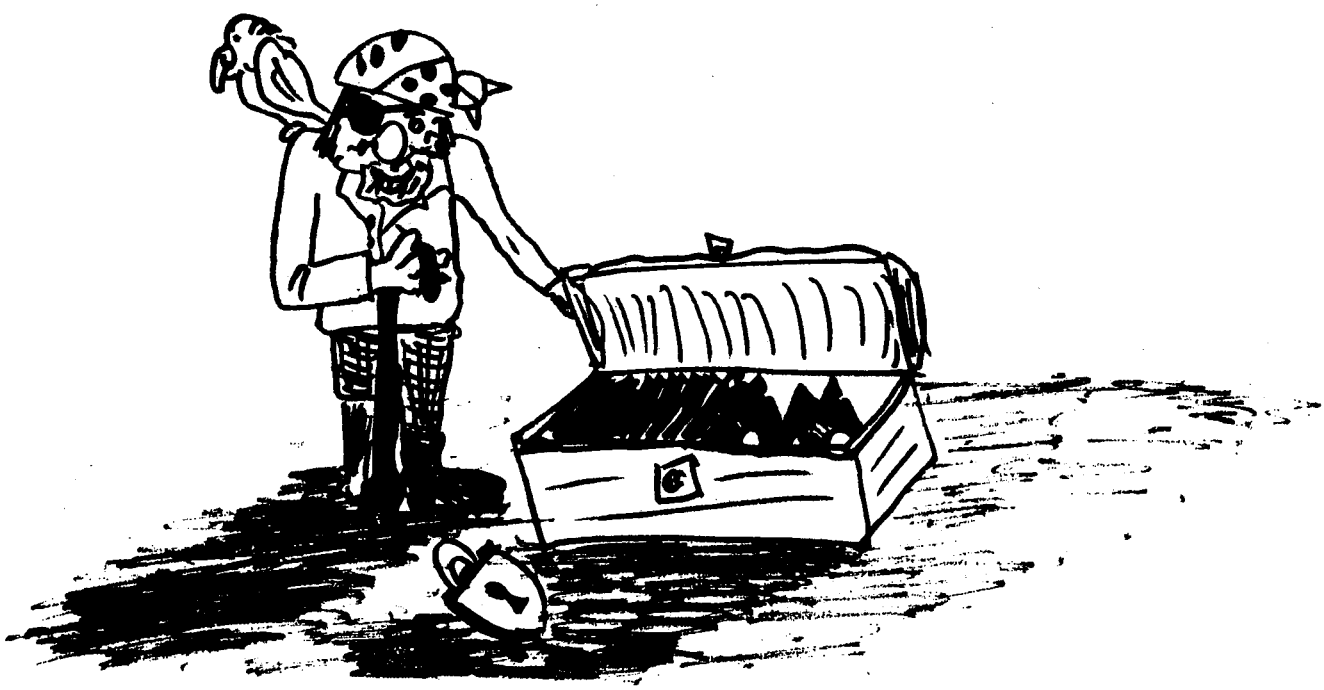
32

Ein Tip: An den Selbstbedienungsständen der Rundfunk- und TV-Händler findet man oft eine große Auswahl an Verbindungskabeln. Zur Kombination deutscher und japanischer HiFi-Geräte gibt es dort meist ein preiswertes Kabel, an dessen einem Ende sich ein 5-poliger Diodenstecker befindet, und am anderen Ende zwei oder vier Cinch-Stecker. Derartige Kabel lassen sich auch für den ATARI-Anschluß verwenden.

SOFTWARE-SCHUTZ

Vor mehr als zehn Jahren ist einer inzwischen legendären Band folgendes Mißgeschick passiert: an das Ende eines ihrer Hits hängten sie eine Klangkollage, die auch einen sekundenkurzen Auszug aus einem Musikstück der 40er Jahre enthielt. Während der Komponist des zitierten Musikstücks zu diesem Zeitpunkt längst verstorben war, hatten die Erben seines geistigen Gutes nichts Eiligeres zu tun, als jene Band auf Schadensersatz in schwindelnder Höhe zu verklagen - und sie bekamen Recht!

Während seit Urzeiten in wenigen Stunden zusammengeschusterte Schlager urheberrechtlich bestens abgesichert sind, sieht es mit dem juristischen Schutz hochgeistiger Software-Produkte auch nach mehr als drei Jahrzehnten EDV noch immer ärmlich aus. Um seine Werke vor unberechtigtem Zugriff zu bewahren, sollte der Programmierer daher möglichst selbst versuchen, es den 'Piraten' möglichst schwer zu machen...



● Wer sich nicht in die Karten sehen lassen will, kann mit POKE 4226,n die DIRECTORY, also das Inhaltsverzeichnis seiner Diskette verlegen. Man gehe dazu wie folgt vor...

1. Diskette formatieren und mit 'H' DOS draufschreiben.
2. Mit 'B' zurück ins BASIC gehen
3. Das zu schützende BASIC-Programm laden bzw. eintippen.
4. Z.B. POKE 4226,12 eingeben.
5. Programm mit SAVE (bzw. LIST)"D:NAME abspeichern.

Schauen Sie nun im DOS mit 'A' in die Directory: wie Sie sehen, sehen Sie nichts! Zumindest nicht den Namen oder irgendeine andere Spur von Ihrem

Programm! Wie fast schon zu erwarten, kann man auch in BASIC das Programm nicht ohne weiteres laden. Selbst wer den Dateinamen Ihres Werkes weiß, bekommt lediglich 'ERROR 170' präsentiert. Der Trick: um das Programm laden zu können, muß man zunächst wiederum die Lokalisierung des Inhaltsverzeichnisses mit POKE 4226,12 bzw. einem anderem (auch fürs Abspeichern) gewählten Wert korrigieren. Nur dann geht's!! Die somit benötigte 'Geheimzahl läßt sich beipielsweise im AUTORUN.SYS oder in einer zuvor aufgerufenen Schlüsselwort-Routine verstecken. Der korrekte Wert in 4226 ist übrigens 105. Er muß stets restauriert werden, wenn man vom BASIC aus ins DOS gehen will; andernfalls ist 'DUP-SYS' nicht auffindbar...

● Zusätzlich sollte man das vor fremden Blicken zu bewahrende Programm natürlich noch mit einem LISTSCHUTZ versehen:

```
32767 POKE PEEK(138)+256*PEEK(139)+2,0:SAVE"D:NAME":NEW
```

Nachdem man diese Zeile an sein Programm angehängt und den gewünschten Dateinamen eingesetzt hat, startet man mit GOTO 32767. Daraufhin wird sofort eine Programmversion auf Diskette geschrieben, die sich nur noch mit RUN starten, jedoch nicht mehr LISTen oder anderweitig manipulieren läßt. Wird dies trotzdem versucht, 'hängt sich der Rechner auf'. ACHTUNG: Unbedingt anderen Dateinamen als fürs Original bzw. andere Diskette verwenden - andernfalls verlieren auch Sie den Zugriff zu Ihrem Werk!

Natürlich können auch Cassetten-Benutzer diesen Listschutz verwenden. Statt SAVE"D:..." heißt es dann SAVE"C:". Auf diese Weise geschützte Software läßt sich dann nur noch mit RUN"C:" starten.

Um unbedarfte Benutzer vor unbeabsichtigtem 'Abstürzen' des Computers zu bewahren, sollte man sicherheitshalber die BREAK-Taste außer Funktion setzen sowie die RESET-Taste auf erneutes Laden programmieren (siehe Abschnitt: 'MEMORY MAP oder WAS LIEGT WO?').

● Will man seinen Mitmenschen nur das Lesen des Programms erschweren, bietet sich folgender VARIABLENSCHUTZ an:

```
32767 FOR A=PEEK(130)+256*PEEK(131) TO PEEK(132)+256*PEEK(133):POKE
A,n:NEXT A
```

Man ergänzt diese Zeile zu seinem Programm und startet mit GOTO. Da die Variablen-Tabelle dann vom Anfang (130,131) bis zum Ende (132,133) mit einunddemselben Zeichen aufgefüllt wird, ist ein mit vielen Variablen behaftetes Programm kaum noch veständlich. Setzt man für n den Wert 61 ein, erhält man anstelle der Variablen Gleichheitszeichen, 42 ergibt Sternchen usw. n=155 ergibt Zeilenvorschübe; variablenreiche Programme ergeben dann herrlich lange Listings... Doch Vorsicht: es gibt bereits Programme, mit deren Hilfe diese Schutzmaßnahme rückgängig gemacht werden kann!

SCHLÜSSELWORT-ROUTINEN

Um ein Programm vor Mißbrauch zu schützen, kann man dessen korrekte Funktion auch von der Eingabe eines Schlüsselwortes abhängig machen. Dies kann auf sehr verschiedene Wege geschen. Hier ein Beispiel:

```
05 REM SCHLUESSELWORT-ABFRAGE
10 DIM WORT$(20):GEHEIM=6
20 PRINT"Schluesselwort?";:INPUT WORT$
30 IF LEN(WORT$)<>6 THEN 32000
40 FOR I=1 TO GEHEIM -
50 SUMME=SUMME+ASC(WORT$(I))
60 NEXT I
70 IF SUMME<>290 THEN 32000
80 REM HIER STARTET IHR WERTVOLLES PROGRAMM
   .
   .
32000 NEW
```

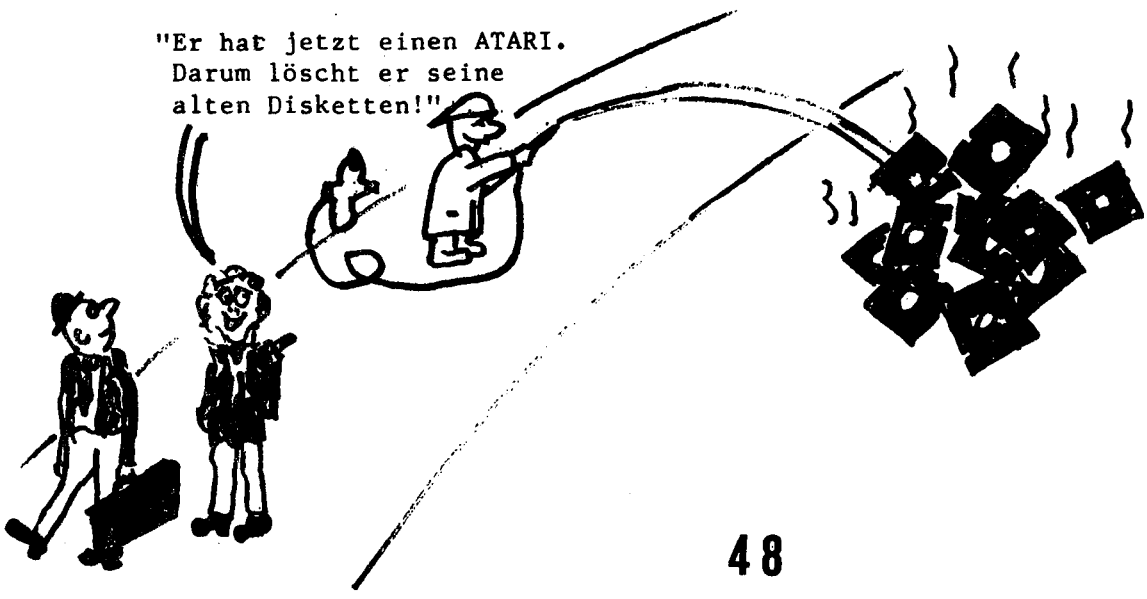
Die Funktionsweise dieser Routine:
Unser Beispiel-Schlüsselwort lautet 'SICHER'. Nachdem in Zeile 20 ein Wort eingegeben wurde, wird zunächst die Zahl der Buchstaben kontrolliert. Hier werden bereits alle Wörter ausgesiebt, die nicht die korrekte Länge aufweisen. Anschließend werden die ASCII-Werte der eingegebenen Buchstaben addiert. Stimmt die Summe nicht mit der Vorgabe (290) überein, erfolgt spätestens hier ein Sprung zu Zeile 32000. Das harmloseste, was man dort hineinschreiben kann, ist 'NEW'. Mit XIO 254,=1,0,0,"D1:" läßt sich die Programmdiskette auch frisch formatieren! (Um zu verhindern, daß gewitzte Leute den Write-Protect-Schlitz überkleben, läßt man in diesem Falle zunächst mit OPEN"D1... überprüfen, ob sich die Diskette beschreiben läßt und sieht für ERROR144 einen Absturz des Programms vor...). So tückisch können Programmierer sein!

Wichtig ist es natürlich, daß das Programm darüberhinaus mit dem bereits vorgestellten List- und/oder Ladeschutz versehen wird. Aus diesem Grunde ist in unserem Beispiel in Zeile 40 auch nicht die Zahl 6, sondern eine Variable verwendet worden.

Falls sich jemand die Mühe machen will, Ihr Schlüsselwort durch Probieren zu finden, muß er die Diskette bzw. Cassette wahrscheinlich so oft laden, bis er durch die Magnetschicht hindurchschauen kann!

DEFEKTE SEKTOREN?

Für die Besitzer der Diskettenstation ATARI 810 bietet sich darüberhinaus noch eine zusätzliche Möglichkeit, sich vor dem 'Software-Klau' zu schützen: siehe Abschnitt 'DISK': 'Sektor-Killer'.



DIE MEMORY MAP ODER 'WAS LIEGT WO IM ATARI?'

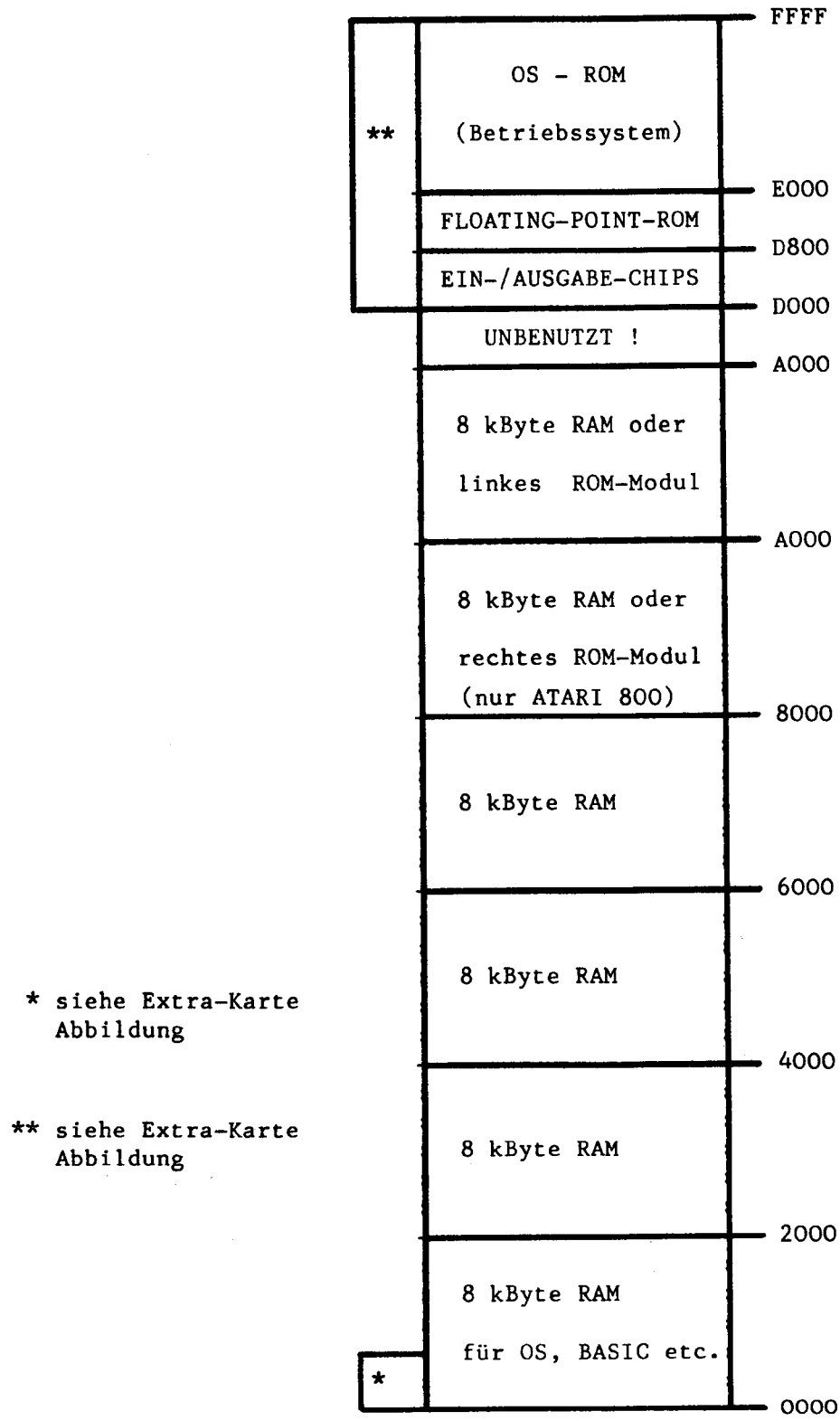


Abbildung 33: Speicher-Karte ('Memory Map') des ATARI-Computers

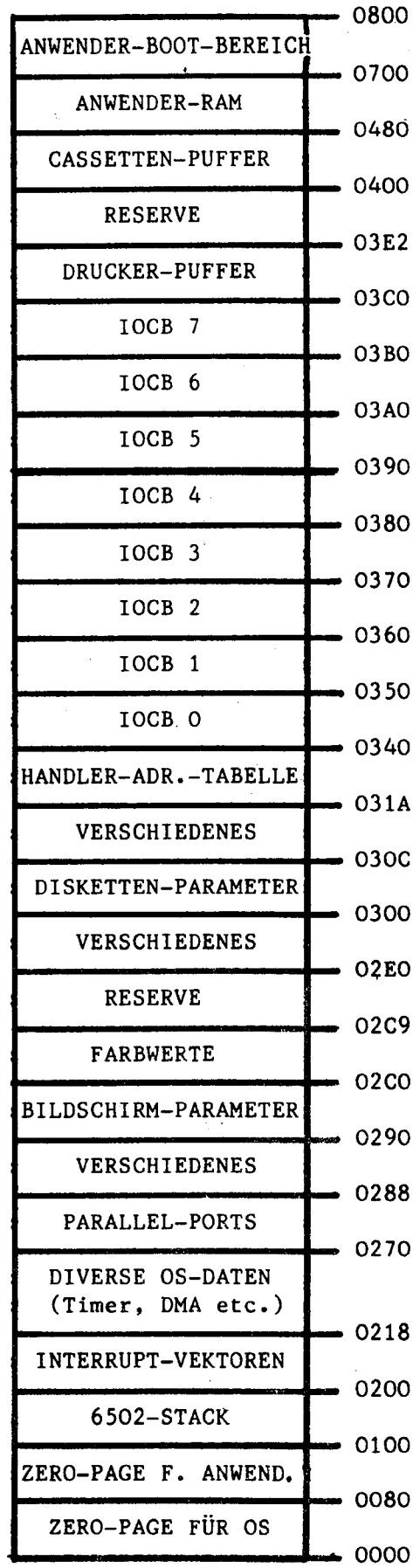
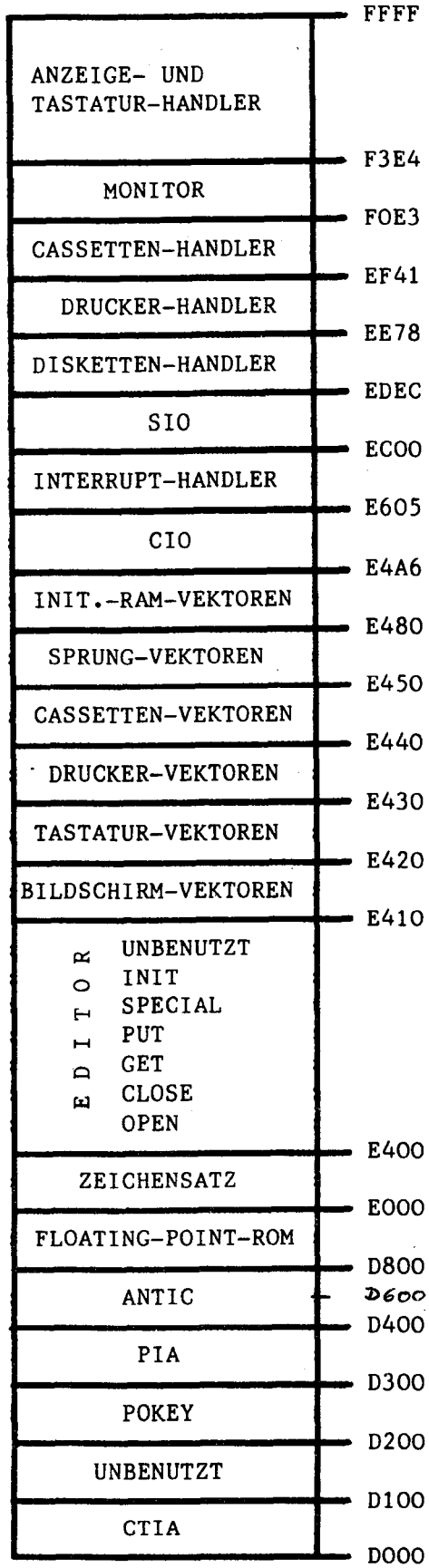


Abbildung 34

Speicherkarten-Auszug

von 0000 bis 0800



Speicheraufbau XL

0000 – 4FFF	RAM	00000 – 20479
5000 – 57FF	RAM- oder Selbsttest	20480 – 22527
5800 – 7FFF	RAM	22528 – 32767
8000 – 9FFF	RAM- oder ROM-Modul	32768 – 40959
A000 – BFFF	RAM- oder ROM-Modul	40960 – 49151
C000 – CFFF	OS oder RAM	49152 – 53247
D000 – D0FF	GTIA	53248 – 53503
D100 – D1FF	Reserviert	53504 – 53759
D200 – D2FF	POKEY	53760 – 54015
D300 – D3FF	PIA	54016 – 54271
D400 – D4FF	ANTIC	54272 – 54527
D500 – D5FF	Jeder Zugriff auf eine dieser Adressen übergibt die ROM-Modul-Kontrolle an das ROM-Modul Interface	54528 – 54783
D600 – D7FF	Reserviert	54784 – 55295
D800 – FFFF	OS oder RAM	55296 – 65535

Abbildung 35

Speicherkarten-Auszug

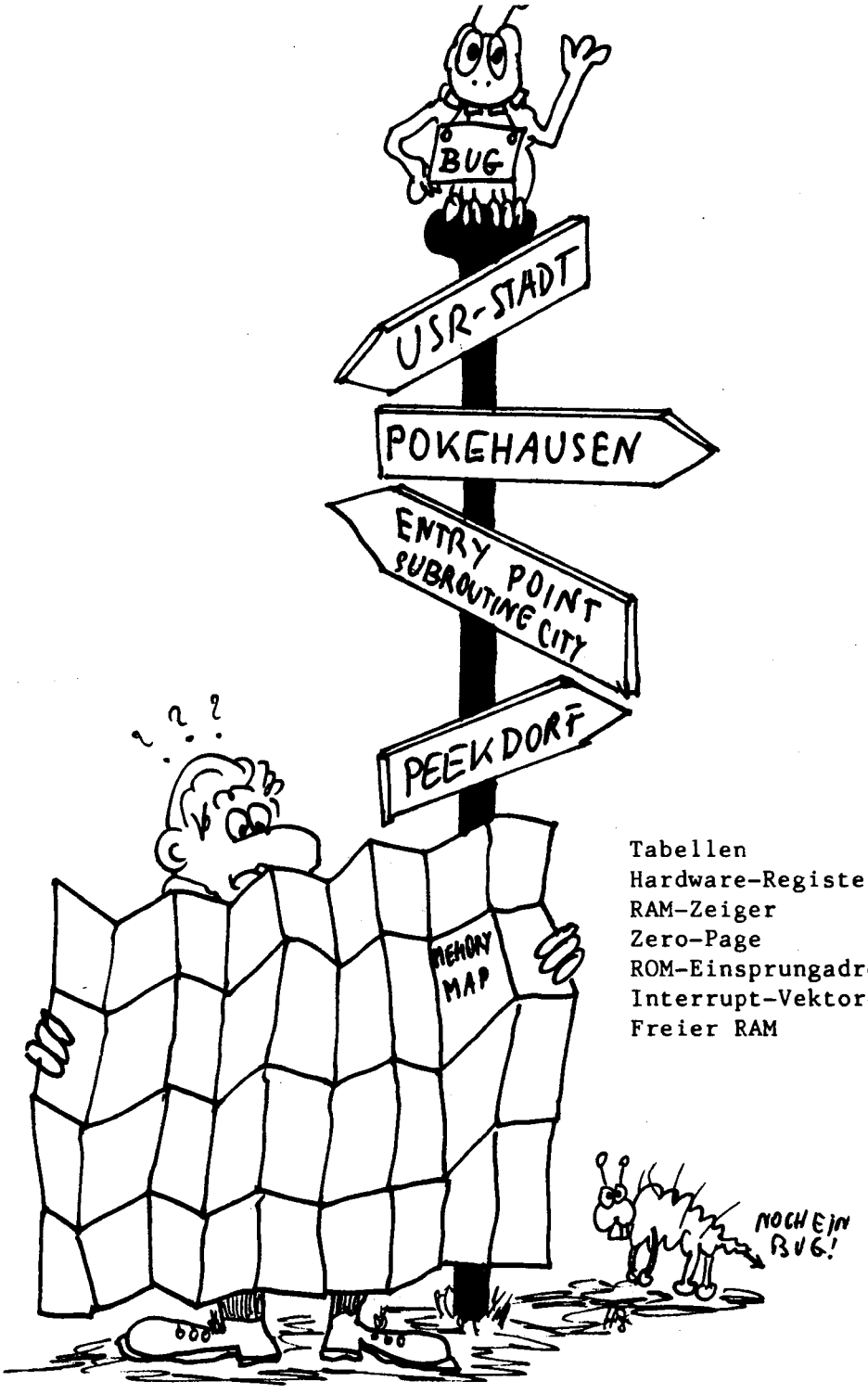
von DOOO bis FFFF

BASIC - HEXDUMP

SPEICHER ANSCHAUEN - ABER ÜBERSICHTLICH!

```
100 REM HEXDUMP *** ABBRECHEN MIT LEERTASTE / ANHALTEN MIT CTRL-1 ***
110 DIM BYT$(10),BB$(4),CH$(40):GOTO 290
120 BB=ASC(BB$)-48
130 IF BB>9 THEN BB=BB-7:IF BB>15 THEN GOTO 150
140 RETURN
150 ? :? :? " ** UNZULAESSIGE EINGABE **":? " ** NUR  H E X  - ZAHLEN **":FOR WAIT=1 TO 500:NEXT WAIT:RUN
160 IF QQ>0 THEN GOTO 150:IF RR>7 THEN GOTO 150
170 RETURN
180 IF QQ>0 THEN GOTO 150
190 RETURN
200 FOR I=1 TO 4
210 OPEN #1,4,0,"K:":GET #1,AA:CLOSE #1
220 BYT$(I)=CHR$(AA):? CHR$(AA);
230 IF ASC(CHR$(AA))<48 THEN GOTO 150:IF ASC(CHR$(AA))>57 THEN IF ASC(CHR$(AA))<65 THEN 150
240 IF ASC(CHR$(AA))>70 THEN 150
250 NEXT I:?
260 BB$=BYT$(1):GOSUB 120:QQ=BB:BB$=BYT$(2):GOSUB 120:RR=BB
270 BB$=BYT$(3):GOSUB 120:DD=BB:BB$=BYT$(4):GOSUB 120:HH=BB
280 CC=QQ*16+RR:FF=DD*16+HH:RETURN
290 PQ=0:GRAPHICS 0:? " LISTE DES SPEICHERINHALTS:":? :? "VON(HEX, VIERSTELLIG): ";:GOSUB 200
300 AB=CC*256+FF:? :? "BIS(HEX, VIERSTELLIG): ";:GOSUB 200
310 BC=CC*256+FF:? :X=0
320 FOR L=AB TO BC:X=X+1
330 C=PEEK(L):GOSUB 560
340 IF X=1 THEN GOSUB 400
350 GOSUB 470
360 IF X=8 THEN X=0:GOSUB 600
370 GOSUB 540
380 NEXT L
390 PRINT :PRINT "FERTIG":END
400 H=INT(L/4096):I=INT((L-H*4096)/256)
410 K=INT((L-H*4096-I*256)/16)
420 M=L-H*4096-I*256-K*16
430 O=H:GOSUB 450:O=I:GOSUB 450:O=K:GOSUB 450:O=M:GOSUB 450
440 ? " ";:RETURN
450 IF O<10 THEN ? O;:RETURN
460 IF O>9 THEN ? CHR$(O+55);:RETURN
470 IF C=0 THEN ? "00";" ";:RETURN
480 D=INT(C/16):E=C-(D*16)
490 IF D<10 THEN ? D;:GOTO 510
500 IF D>9 THEN ? CHR$(D+55);
510 IF E<10 THEN ? E;:GOTO 530
520 IF E>9 THEN ? CHR$(E+55);
530 ? " ";:RETURN
540 IF PEEK(764)=255 THEN RETURN
550 PRINT :PRINT "ABGEBROCHEN":END
560 XK=C:IF XK>154 AND XK<160 THEN XK=46
570 IF XK<33 OR XK=125 THEN XK=46
580 IF XK=127 OR XK>252 THEN XK=46
590 CH$(X)="":CH$(X)=CHR$(XK):RETURN
600 ? CH$(1):RETURN
```

Mit diesem Programm können Sie beliebige Speicherinhalte - egal ob ROM oder RAM - ausdrucken! Nach Angabe der (hexadezimalen) Start- und Endadresse bekommen Sie pro Bildschirmzeile acht Bytes ausgedruckt, und zwar sowohl hexadezimal als auch in ASCII-Repräsentation (incl. Invers- und Graphikzeichen!).



Tabellen
Hardware-Register
RAM-Zeiger
Zero-Page
ROM-Einsprungadressen
Interrupt-Vektoren
Freier RAM

Teil 2:

DIE MEMORY MAP ODER: WAS LIEGT WO IM ATARI ?

Auf den folgenden 20 Seiten finden Sie eine Zusammenstellung aller wichtigen Adressen Ihres ATARI-Computers. Vieles läßt sich für eigene Programme ausnutzen. Die aufgelisteten Adressen wurden aus diversen Manuals, Source-Listings und eigenen Experimenten zusammengetragen und werden z.T. anhand von Beispielen erläutert. Aus Platzgründen ist dies leider nicht für sämtliche wichtigen Speicherzellen möglich. Die Auflistung geschieht in der Reihenfolge:

NAME (Original-Listing) / Dezimale Adresse(n) / Hexadezimale Adresse(n)

CASINI 2,3 2,3
Zeiger für Cassetten-Boot-Initialisierung. War das 'Booten' erfolgreich, erfolgt ein JSR zur hier angegebenen Adresse.

RAMLO 4 4
RAM-Zeiger für den Speichertest

TRAMSZ 6 6
Zeitweiliges Register für die zur Verfügung stehende RAM-Größe

TSTDAT 7 7
Datenregister für den RAM-Test

WARMST 8 8
Warmstart-Kennung (=-1, wenn RESET gedrückt wurde und =0 bei erfolgreicher Einschalt-Initialisierung). Anwenderprogramme u. Daten bleiben bei Sprung zu dieser Adresse erhalten.

BOOT 9 9
Boot-Kennung.
Bit1 gesetzt: auf RESET Sprung zu DOSINI (Disketten-Initialisierung)
Bit2 gesetzt: auf RESET Sprung zu CASINI (Cassetten-Initialisierung)
Sind Bit1+2 gesetzt, erfolgt zuerst Cassetten-Initialisierung.

DOSVEC 10,11 A,B
Start-Zeiger für Diskettenprogramme

DOSINI 12,13 C,D
Wird zur Ablage der Adresse verwendet, bei der nach Laden des DOS das Anwenderprogramm startet. Mit einem indirekten JSR hierher läßt sich die Anwendersoftware initialisieren.

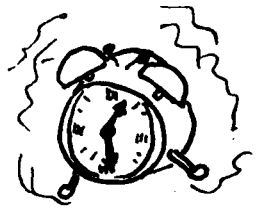
APPMHI 14,15 E,F
Enthält die höchste Adresse des von Anwender nutzbaren RAM-Speichers.

POKMSK 16 10
Pokey-Interrupt-Zeiger. Enthält Kennbits für die verschiedenen Quellen, die einen Interrupt auslösen können:

Bit7=Interrupt-Freigabe für BREAK-Taste
Bit6=Interrupt-Freigabe für übrige Tasten
Bit5=Interrupt-Freigabe für 'Serielle Eingangsdaten bereit'-Meldung
Bit4=Interrupt-Freigabe für 'Serielle Ausgangsdaten erforderlich'-Meldung
Bit3=Interrupt-Freigabe für 'Serielle Ausgabe beendet'-Meldung
Bit2=Interrupt-Freigabe für Timer 4
Bit1=Interrupt-Freigabe für Timer 1

Damit läßt sich allerlei verriegeln! Z.B. bewirkt POKE16,64 in Verbindung mit POKE 53774,64 eine Blockierung der Break-Taste POKMSK ist Schattenregister zu IRQEN=53774).

BRKKEY 17 11
Abfrage der BREAK-Taste. 0=Taste=gedrückt worden; anderer Wert=Taste nicht gedrückt.



RTCLOCK 18,19,20 12,13,14
Interne Uhr!
20: zählt im 50-Hz-Takt von 00 bis FF
19: wird um eins erhöt, wenn 20 wieder bei 00 zu zählen beginnt.
18: wird um eins erhöht, wenn 19+20 auf 00 gehen. - Auch 18 und 19 zählen von 00 bis FF
ACHTUNG! Amerikanische Programme, die diese Uhr benutzen, sind auf die dort verwendeten 60-Hz-Maschinen ausgelegt! In vielen US-Programmen ist daher als Faktor 5/6 zu ergänzen.

BUFADR 21,22 15,16
Register für die indirekte Pufferadresse. Wird als zeitweiliger Zero-Page-Zeiger auf den momentanen Disketten-Puffer (128 Bytes) benutzt.

ICCOMT 23 17
Vektor-Befehl

DSKFMS 24,25 18,19
Disk-FMS-Zeiger(FMS = Dateiverwaltungssystem)

DSKUTL 26,27 1A,1B
DUP-Zeiger (DUP = Disketten-Hilfsprogramm-Paket)

PTIMOT 28 1C
Drucker-Zeitabschaltung (TIMEOUT).

PBPNT 29 1D
Zeiger auf Drucker-Puffer. Zeigt im Drucker-Puffer von 0 bis zum in PBUFSZ festgelegten Höchstwert.

PBUFSZ 30 1E
Länge des Drucker-Puffers im jeweiligen Modus:
Normal = 40 Bytes
Doppelte Breite = 20 Bytes

PTEMP 31 1F
Der Drucker-Handler verwendet dieses zeitweilige Register, um den Wert des Zeichens zu speichern, das zum Drucker ausgegeben werden soll.

IOCBs (Ein-/Ausgabesteuerungsblöcke

++ IOCBs in der Zero-Page ++

ZIOCB 32 20
'Handler-Index-Nummer'. Wird vom Betriebssystem als ein Index auf die Dateinamentabelle für eine geöffnete Datei gesetzt. Wird auf 255 ($FF) gesetzt, wenn keine Datei geöffnet ist.

ICDNOZ 33 21
Diskettenlaufwerk-Nummer. Wird vom Betriebssystem für das zu benutzende Laufwerk auf 1 bis 4 gesetzt.

ICCOMZ 34 22
Befehls-Byte aus dem Anwenderprogramm, mit dem festgelegt wird, wie der Rest des IOCBs zu formatieren ist.

ICSTAZ 35 23
Status-Byte, das für das angesproche Gerät (Device). Wird vom Betriebssystem gesetzt. Muß nicht dem Wert beim STATUS Befehl entsprechen.

ICBALZ/H 36,37 24,25
Pufferadresse für den Datentransfer oder Adresse des Dateinamens für Befehle wie OPEN, STATUS, etc.

ICPTLZ/HZ 38,39 26,27
Adresse der Put-Byte-Routine minus 1. Wurde kein Gerät geöffnet, zeigt dieser Vektor auf die Routine "IOCB nicht offen".

ICBLLZ/HZ 40,41 28,29
Pufferlängen-Zähler für PUT/GET-Operationen. Wird bei jedem übertragenen Byte um 1 erniedrigt.

ICAX1Z 42 2A
Hilfsinformation, die beim Öffnen benutzt wird, um die Art des Dateizugriffs festzulegen.

ICAX2Z 43 2B
CIO-Arbeitsvariablen. Einige Funktionen der seriellen Schnittstelle benutzen dieses Byte.

ICAX3&4Z 44,45 2C,2D
Wird bei NOTE und POINT für die Disketten-Sektornummern benutzt, kann jedoch auch andere Funktionen erfüllen.

ICAX5Z 46 2E
Das Byte in obigem Sektor (IOCBx16).

ICAX6Z 47 2F
Zeichen-Byte für die Laufende Operation.

++ Weitere IOCBs liegen im Bereich von 832 bis 959
++ Der Aufbau geschieht nach demselben Schema
++ ACHTUNG: Zero-Page-IOCBs nicht für normale E/A benutzen!

STATUS 48 30
Interne Status-Speicherung

CHKSUM 49 31
Einzelbyte-Summe mit Übertrag auf das niederwertige Bit.

BUFRLO 50 33
Zeiger auf das niederwertigere Byte des Daten-Puffers.

BUFRHI 51 34
Zeiger auf das höherwertige Byte des Daten-Puffers.

BFENLO 52 35
Nächstes Byte (Low Byte) nach dem Ende des Puffers.

BFENHI 53 36
Nächstes Byte (High Byte) nach dem Ende des Puffers.

CRETRY 54 37
Anzahl der Wiederholungen, wenn die Ausführung eines Befehls scheitert.

DRETRY 55 38
Anzahl der Wiederholungen, wenn das Ansprechen eines Gerätes scheitert (Z.B. Disk)

BUFRFL 56 39
Kennung für 'Puffer voll'.

RECVDN 57 3a
Kennung für 'Datentransfer zur Peripherie fertig'.

XMTDON 58 3B
Kennung für 'Datentransfer von der Peripherie fertig'.

CHKSNT 59 3C
Kennung für 'Prüfsumme gesendet'.

BPTR 61 3D
Index zu den Daten des Datensatzes, der gerade geschrieben bzw. gelesen wird. Die hier gefundene Zahl liegt zwischen 0 und dem momentanen Wert in BLIM (28A). Ist BLIM = BPTR, ist der Cassettenpuffer CASBOFF (3FD) entweder ganz leer (nach Lesen) oder ganz voll (nach Schreiben).

FTYPE 62 3E
Festlegung des Abstands zwischen den Datenblöcken auf Cassette. Zwei Typen sind möglich: FTYPE=0 ergibt normale, FTYPE=S80 ergibt größere Zwischenräume (der Unterschied zeigt sich z.B. beim Abspeichern mit 'SAVE"C:' bzw. 'CSAVE').

FEOF 63 3F
Kennung des Cassetten-Handlers, um das Ende einer Cassetten-Datei anzuzeigen.

FREQ 64 40
Hier schreibt der Cassetten-Handler hinein, wieviele Signaltöne der Konsolen-Lautsprecher abgeben soll (z.B. Aufnahme: 2x).

SOUNDR 65 41
Kennung für die akustische Anzeige von E/A-Operationen . Setzt man hier 0 ein, geschieht der Disketten- bzw. Cassetten- Datentransfer geräuschlos!

CRITIC 66 42
1 = Ausschalten der Tasten-Wiederholfunktion und Änderung des Summer-Tons bei CTRL-2.

ATRACT 77 4D
Ein Wert kleiner als 128 ist der normale Wert. 128 ergibt eine kleinere Helligkeit und wechselt die Farben um die Bildröhre zu schonen. Wenn ein Programm ohne Benutzung der Tastatur arbeitet und ein Farbwechsel unererwünscht ist, sollten Sie alle 5-7 Minuten eine Zahl-128 in diese Speicherzelle schreiben. Dies ist gilt insbesondere für Anwendungen (sprich Spiele), die ausschließlich mit Steuerknüppeln oder Drehreglern bedient werden.

80 50 Modem serial bus device address

DRKMSK 78 4E
254=Normale Helligkeit

LMRGIN 82 52
Linker Bildschirmrand. Ist normalerweise auf 2 gesetzt.

RMRGIN 83 53
Rechter Bildschirmrand. Ist normalerweise auf 39 gesetzt.

Der Bildschirm kann also bei Bedarf noch etwas verbreitert werden (82=0 u. 83=39).

ROWCRS 84 54
Reihe auf dem Bildschirm, in dem sich der Grafikcursor gerade befindet.

COLCRS 85,86 55,56
Spalte auf dem Bildschirm, in der sich der Cursor gerade befindet.

Diese zwei Adressen können also dazu benutzt werden, den Cursor an eine bestimmte Stelle zu setzen bzw. seine Position auf dem Bildschirm zu ermitteln.

CRMODE 87 57
Wird dazu benutzt, das Betriebssystem 'auszutricksen', sodaß es glaubt, ein anderer Grafikmodus als der ursprüngliche sei eingeschaltet. Diese Adresse ist auch beim Erstellen von Display-Listen wichtig, um die normale Anwendung von PLOT- und PRINT- Befehlen zu ermöglichen. Wird mit Werten von 0 bis 8 (wie die normalen Grafikstufen) belegt.

SAVMSC 88,89 58,59
Niedrigste Adresse des Bildschirmspeichers. Die Daten in dieser Adresse werden in die obere linke Bildschirmecke geschrieben und die nächste Anzahl von Bytes darauffolgend.

OLDROW 90 5A
Vorherige Reihe des Grafik-Cursors.

OLDCOL 91,92 5B,5C
Vorherige Spalte des Grafik-Cursors.

OLDCHR 93 5D
Daten unter dem Cursor. Wird vom Betriebssystem bei der Cursor-Steuerung verwendet. Anschaulich wird dies, wenn Sie in BASIC folgendes ausprobieren:
POKE 93,219:PRINT"- ";

OLDADR 94 5E
Enthält die Speicheradresse der momentan sichtbaren Cursor-Position.

NEWROW 96 60
Punkt bzw. Reihe, zu der DRAWTO gehen soll.

NEWCOL 97,98 61,62
Punkt bzw. Spalte, zu der DRAWTO gehen soll.

RAMTOP 106 6A
Höchste Adresse des RAM- Bereichs. Angegeben als Anzahl der Pages (eine

121,122 79,7A Anfangsadresse Tastenfeld XL
Reihenfolge: 64 Bytes klein, 64 Groß, 64 CTRL
SHIFT, CTRL, CTRL-1 nicht umlegbar

Page= 256 Bytes). Durch Abänderung dieses Vektors läßt sich gegen anderweitige Benutzung geschützter Speicherplatz für Maschinenprogramme reservieren. Und zwar mit Hilfe des Befehls POKE 106,n (n ist PEEK(106) abzüglich der Zahl der 256-Byte-Seiten, die geschützt werden sollen).
! ++ Der Rest der Zero-Page steht für Anwenderprogramme ++ !
++ bzw. für BASIC zur Verfügung. Es folgt die Auflistung der neun BASIC-Zeiger ++

LOMEM 128,129 80,81
Token-Ausgabe-Puffer, d.h. der Puffer, den ATARI-BASIC dazu benutzt, eine Befehlszeile in die vom Interpreter verständlichen Kurz-Codes (Tokens) zu übersetzen. Dieser Puffer liegt am Ende des zur Verfügung stehenden RAMs.

VNTP 130,131 82,83
Liste der Variablennamen; eine Liste aller Variablennamen, die im Programm verwendet wurden. Sie werden als ATASCII-Zeichen gespeichert. Die Namen werden in der Reihenfolge gespeichert, in der sie eingegeben werden. Die Variablen werden dabei in drei Gruppen eingeteilt:

(1) Numerische Variablen - Beim letzten Buchstaben des Namens ist das höchstwertige Bit (MSB) gesetzt.
(2) Text-(String-)Variablen - Das letzte Zeichen ist ein 'S', bei dem das MSB gesetzt ist.
(3) Matrix-Variablen - Das letzte Byte ist ein '('-Zeichen, dessen MSB gesetzt ist (MSB=Bit7, höchstwertiges Bit).
Display-Liste

VNTD 132,133 84,85
(Blindes) Ende der Variablennamen-Tabelle. ATARI-BASIC benutzt diesen Zeiger, um das Ende dieser Tabelle anzuzeigen. Wird kein Variablenname benutzt, deutet er auf ein blindes Null-Byte (Dummy). Werden 128 Variablen benutzt, zeigt er auf das letzte Byte des letzten Variablennamens.

VVTP 134,135 86,87
Variablen-Wertetabelle; Diese Tabelle enthält die aktuelle Informationen für jede Variable. Dabei sind für jede Variable in der Variablentabelle 8 Bytes reserviert, aus denen Variablenart (s.o.), Dimensionierung etc. hervorgehen.

STMTAB 136,137 88,89
Adresse der Anweisungs-Tabelle; diese Tabelle enthält alle eingegebenen und in TOKENs umgewandelten Befehle.

STMCUR 138,139 8A,8B
Laufende Anweisung. Dieser Zeiger wird von ATARI-BASIC dazu benutzt, innerhalb einer Zeile für bestimmte Tokens eine Referenz zur Anweisungstabelle herzustellen. Wartet BASIC auf eine Eingabe, deutet der Zeiger auf den Anfang der Direkteingabe-Zeile.

STARP 140,141 8C,8D
Speicherbereich für Text und Arrays; Dieser Block enthält alle Text- und Felddaten. Zeichen werden dabei in ATASCII gespeichert, d.h. jedes Zeichen belegt ein Byte. Arrays benötigen 6 Bytes für pro Zahl. Der zur Verfügung stehende Speicherplatz schrumpft daher mit zunehmender Dimensionierung.

RUNSTK 142,143 8E,8F
Run Time Stack (Ablauf-Stapelspeicher); dieser Software-Stack enthält Eintragungen für alle in Arbeit befindlichen GOSUB- und FOR/NEXT-Anweisungen.

MEMTOP 144,145 90,91
Von der hier festgelegten Adresse bis zum Anfang der Display-Liste kann Ihr ATARI-BASIC-Programm reichen. Auch die FRE- Funktion orientiert sich an diesem Zeiger (HIMEM (2E5,2E6) minus MEMTOP).
ACHTUNG: MEMTOP im ATARI-BASIC ist nicht identisch mit dem im Betriebssystem mit MEMTOP benannten Zeiger!

+++++

STOPLN 186,187 BA,BB
Zeilennummer, in der ein STOP oder TRAP aufgetreten ist.

ERRSAVE 195 C3
Fehlernummer, die den STOP oder TRAP ausgelöst hat.

PTABW 201 C9
Schrittweite für TAB. Normalerweise auf 10 gesetzt.

203-209 CB-D1 In BASIC nicht belegt

FRO 212-217 D4-DB
Fließkommaregister 0 (Der Wert der von der USR-Funktion zurückgegeben wird.)
nur 212/213 = D4/D5

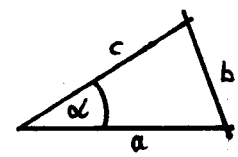
FR1 224-229 E0-E5
Fließkomma-Register 1

FLPTR 252,253 FC,FD
Pointer To a FP-number

CIX 242 F2
Zeichen-Index. mit INBOFF benutzt

INBUFF 243,244 F3,F4
Zeiger zum Texteingabe-Puffer.

RADFLG& 251 FB
RAD/DEG Schalter. RAD=0(Bogenmaß), DEG=6(Grad)



VDSLST 512,513 200,201
Interrupt-Vektor der Display-Liste (DLI). DLIs werden dazu benützt, den normalen Fluß der Ausführung der Display-Liste anzuhalten, um für einige Mikrosekunden etwas anderes zu machen. So ist es z.B. möglich, Musik unabhängig vom übrigen Programmgeschehen spielen zulassen.

VPRCED 514,515 202,203
Seriell Proceed (serielle Schnittstelle)

VINTER 516,517 204,205
Serieller Interrupt.

VBREAK 518,519 206,207
Vektor zur BREAK-Anweisung.

VKEYBD 520,521 208,209
Vektor zum Tastatur-Interrupt.

VSERIN 522,523 20A,20B
Serielle Schnittstelle: bereit zum Empfang von Daten.

VSEROR 524,525 20C,20D
Serielle Schnittstelle: bereit zum Senden.

VSEROC 526,527 20E,20F
Serielle Schnittstelle: Übertragung abgeschlossen.

POKEY timer

VTIMR1 528,529 210,211
Vektor von Timer 1.

VTIMR2 530,531 212,213
Vektor von Timer 2.

VTIMR3 532,533 214,215
Vektor von Timer 3.

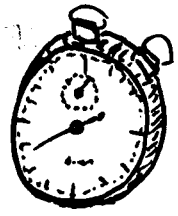
VIMIRQ 534,535 216,217
Unmittelbarer IRQ-Zeiger (generell)

++++++++

STOP-

++ Uhren in Ihrem ATARI: die Timer ++

Diese Timer zählen jede 1/50stel (NICHT 1/60!!) Sekunde um eins herunter, bis sie 0 erreicht haben. Man kann sie für Zeitmessungen, unabhängig von der Programmausführung verwenden und von ihnen Interrupts auslösen lassen. Siehe unser Beispiel 'HELLIGKEITSSTEUERUNG'!).



CDTMV1 536,537 218,219
Wert für Timer 1

CDTMV2-5 538-545 21A-221
Wert für den Timer 2 bis 5.

VVBLKI 546,547 222,223
Vertical Blank Immediate. Hier nur kurze Interrupt-Routinen einbauen (bis zu 3000 Maschinenzyklen), da von hier aus zeitkritische E/A stattfindet.

VVBLKD 548,549 224,225
Vertical Blank Deferred. Von hier aus können eigene Interrupt-Routinen bis zu einer Länge von 25000 Maschinenzyklen angesprungen werden.

CDTMA1 550,551 226,227
Sprungadresse für Timer 1.

CDTMA2 552,553 228,229
Sprungadresse für Timer 2.

CDTMF3-5 554-558 22A-22E
Kennung für Timer 3 bis 5.

++++++++

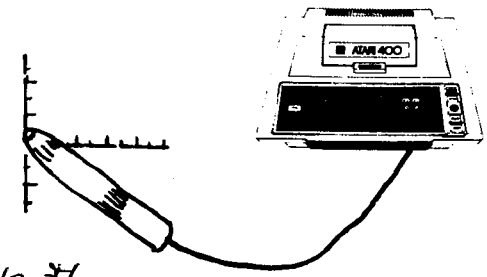
DMA

SDMCTL 559 22F D403
DMA-Steuerung des ANTIV-Chip. ANTIC führt interruptgesteuert die Bildschirmaufbereitung durch, und zwar im direkten Speicherzugriff (DMA). Wird kein Bild gebraucht, können Sie den DMA und damit den Bildschirm vorrübergehend abschalten und auf diese Weise eine bis zu 30% höhere Programmausführungsgeschwindigkeit erzielen! DMA ein = ursprünglicher Wert; DMA aus = 0.

TIP: Probieren Sie einmal POKE 559,47! Sie haben dann mehr als 40 Zeichen pro Zeile, allerdings versetzt. Experimentieren Sie! Zum DMA siehe auch Adresse 54272.

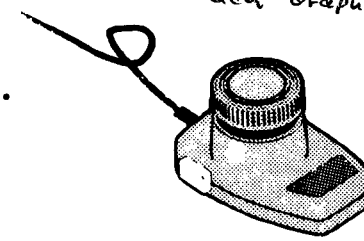
SDLSTL 560,561 230,231 D402/03
Zeiger für die Display-Liste. Gibt die Startadresse der Display-Liste an, die man natürlich auch verändern kann.

SSKCTL 562 232
Steuerung der seriellen Schnittstelle



LPENH 564 234 D40C
Horizontale Position des Lichtgriffels.

LPENV 565 235 D40D
Vertikaler Wert des Lichtgriffels.
DFLAGS 576 240 Disk Boot File Flag
COLDST 580 244
1 = Kaltstart / 0 = Normal.
POKE 580,1 und Drücken der SYSTEm-RESET-Taste entspricht dem erneuten Einschalten des Computers (Kaltstart) 622 26E Finescrolling
0 = normal
255 = Finescroll, danach Graphmod neu eröffnen
GPRIOR
GPRIOR 623 26F
Schattenregister von 53275.

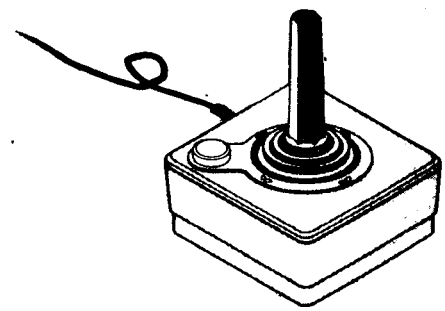


PADDLO 624 270 D200
Wert von Paddle 0 (Bereich von 0 bis 228).

PADDL1-7 625-631 271-277 ...
wie PADDLO für die anderen Drehregler.

STICKO-3 632-635 278-27B D300 ...
Register zur Aufnahme der Bewegungsrichtung der Steuerknüppel 1 bis 3.

PTRIGO-7 636-643 27C-283
Roter Knopf der Drehregler:
0= gedrückt / 1= nicht gedrückt.



STRIGO-3 644-647 284-287 D001 ...
Roter Knopf der Steuerknüppel:
0= gedrückt / 1= nicht gedrückt.
BLIM 650 28A
Siehe BPTR (61)

TXTROW 656 290
Reihe, in der sich der Text-Cursor befindet. (0-3) (Bei Graphik-Modus im Textfenster!)

TXTCOL 657,658 291,292
Spalte, in der sich der Text-Cursor befindet. (0-39)

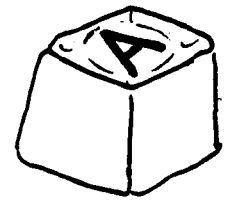
TXTMSC 660,610 294,295
Obere linke Ecke des Textfensters.

TABMAP 675-689 2A3-2B1
Eine 1 in jeder Bit-Position von jedem dieser 120 Bytes verursacht einen Tab-Stop.
Um alle Tabs zu löschen, wird in jedes dieser Bytes der Wert 0 geschrieben.

~~702~~ 2D

725,726 2D5,2D6 Sektorlänge der Disk.

INVFLG 694 2B6
128 = Invers Video (Atari-Taste)
0 = Normal



729 2D9 Verzögerung bis Tastatur-Repeat
730 2DA Repeat-Geschw.
731 2DB =Ø Tastatur-Beep
<>Ø No Beep

SHFLOK 702 2BE
0 = Kleinschrift
64 = Großschrift

BOTSCR 703 2BF
Anzahl der auszudruckenden Textzeilen (24, 4 oder 0).

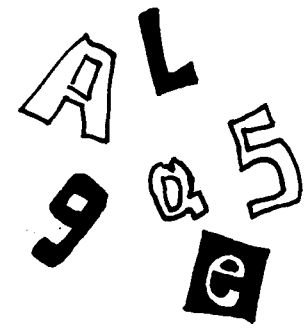
732 2DC HELP
Ø = nicht gedrückt
17 = HELP
81 = SHIFT/HELP
145 = CTRL/HELP
nach Abfrage wieder Ø setzen

PCOLRO-3 704-707 2C0-2C4 D012 ...
Farbe für Player & Missile 0-3.

COLORO 708 2C4 ~~28~~ 40 = $28 D016...
Farbregister 0 (SETCOLOR 0) ;Großbuchstaben

COLOR1 709 2C5 202 = $CA
Farbregister 1 (SETCOLOR 1) ;Kleinbuchstaben

COLOR2 710 2C6 148 = $94
Farbregister 2 (SETCOLOR 2) ;Inverse Großschrift.



COLOR3 711 2C7 70 = $46
Farbregister 3 (SETCOLOR 3) ;Inverse Kleinschrift.

COLOR4 712 2C8
Farbregister 4 (SETCOLOR 4) ;Grafikbereich

MEMTOP 741,742 2E5,2E6 Ø
Zeiger zum höchsten vom Betriebssystem belegten Speicherplatz.
(outer Screen)

MEMLO 743,744 2E7,2E8
Zeiger zum niedrigsten vom Betriebssystem belegten Speicherplatz.

DVSTAT 746 2EA
Kanal-(Geräte-)Status.

CRSINH 752 2F0
0 = Cursor an
1 = Cursor aus
Wo es störend wirkt, läßt sich also mit POKE 752,1 der Cursor unterdrücken!

KEYDEL 753 2F1
0 = keine Taste gedrückt.
3 = eine Taste gedrückt.

CHACT 755 2F3 D401
Register für den Zeichen-Modus.
0 = Inverszeichen als Normalzeichen
1 = bringt Inverszeichen
2 = bringt Normalzeichen
3 = bringt solide Blöcke in Inversdarstellung
4-7 wie 0-3, aber in Spiegelschrift!

CHBAS 756 2F4 D409
Register für die Zeichenbasis: 224 = Großschrift (Default-Wert) 204
226 = Kleinschrift 206 } int. Zeichensätze im XL

ATACHR 763 2FB
Das letzte ATASCII Zeichen, das gelesen oder geschrieben worden ist oder der Wert des Grafikpunktes. Beim DRAWTO-Befehl (als XIO) wird hier auch der Farbwert für den FILL-Befehl abgelegt.

CH 764 2FC
Der interne Code für die letzte gedrückte Taste. 255 löscht. Anschließend können Sie mit PEEK(764) Tastaturwerte (die jeweils zuletzt gedrückte Taste) abfragen, ohne daß die Eingaben mit RETURN abgeschlossen werden müssen.

FILDAT 765 2FD
Farbe für den FILL-Befehl beim Grafik-Betrieb (XIO).

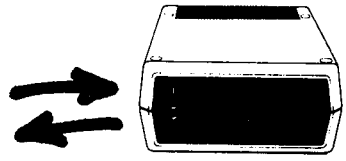
DSPFLG 766 2FE
Display-Kennung:
0 = Normal
1 = Control-Zeichen

SSFLAG 767 2FF
Start-/Stop-Kennung für'Paging', d.h. seitenweises Anschauen von Bildschirminhalten. Wird durch gleichzeitiges Drücken von CTRL und 1 (CTRL-1) gesetzt.

DDEVIC 768 300
Geräte-Bus-Identifikation.Wird nicht vom Anwender benutzt.

DUNIT 769 301
Disketten-Laufwerk-Nummer (1-4).

DCOMND 770 302
Auszuführender Diskettenbefehl, also
S21 Diskette formatieren
S50 PUT Sektor ohne Verify
S52 GET Sektor
S53 GET Status
S57 PUT Sektor mit Verify



DSTATS 771 303
Status-Code des Gerätes. Hier bekommen Sie eine Status-Meldung nach Abschluß der o.a. Operationen.

DBUFLO/H 772,773 304,305
Hier steht die Adresse, ab der die Daten eines Diskettensektors geschrieben bzw. gelesen werden.

DTIMLO 774 306
Zeitabschaltung für den Disk-Handler.

DBYTLO/H 776,777 308,309
Anzahl von Bytes, die bei der zuletzt durchgeführten Operation zur oder von der Disk übertragen wurden. Wird vom Handler gesetzt.

DAUX1/2 778,779 30A,30B
Nummer des Diskettensektors von 1 bis 720, der beschrieben bzw. gelesen wird.

HATABS 794-831 31A-33F
Liste für Handler-Adressen (3 Bytes pro Handler). Maximal sind 12 Eintragungen möglich. Die Liste enthält die aus jeweils einem Buchstaben bestehenden Zeichen für die Geräte-Namen (also K,P,E,S,C,D) und die Handler-Adresse für jede Eintragung. Der Rest der 28 Bytes besteht aus Nullen. - Ein Handler ('Hantierer') ist eine Zusammenstellung von Anweisungen, die dem Computer mitteilen, wie er Bildschirm, Diskette, Drucker etc. handhaben soll.

IOCBO 832-847 340-34F
Kontrollblock 0 für Input/Output.

ICPTL/H 838,839 346,347
Die Eingabe von POKE 838,166 und POKE 839,238 führt dazu, daß alles, was normalerweise auf dem Bildschirm ausgegeben wird, nun an den Drucker geht. Die normalen Werte sind: POKE 838,163 und POKE 839,246. Die Benutzung für andere Geräte ist ebenfalls möglich.

ICAX1 842 34A
13 = Lesen vom Bildschirm
12 = Auf Bildschirm schreiben

IOCB1-7 848-959 350-3BF
E/A-Steuerungs-Blöcke (IOCB) 1 bis 7 (Genauere Beschreibung der Struktur eines IOCBs siehe Zero-Page).

960-999 3C0-3E7 Druckerpuffer

LBUFF 1408 580
Text-Puffer.

1801 709 Anzahl der gleichzeitig zu öffnenden Files (normal: 3)

?????? 1802 70A
Die Anzahl der Diskettenstationen Ihres Systems ist hier gespeichert (normalerweise auf 2 gesetzt). Wenn mehr oder weniger vorhanden sind, sollte die Anzahl hier geändert werden und dann das geänderte DOS wieder auf Diskette gespeichert werden (H-Option im DOS-Menü). Ein Beispiel: Angeschlossene Laufwerke: 1,2 und 4 → 00001011 (Binär) → 11 (Dezimal).

?????? 1913 779
80 = Schaltet Lese-Test beim Schreiben aus.
87 = Schaltet den Test wieder an.
Den Wert $80 Zu benutzen ist wesentlich schneller, bei schlechtem Diskettenmaterial ist jedoch Vorsicht geboten. Bei Verwendung hochwertiger Disketten ist jedoch kein merklicher Anstieg der Fehlerrate festzustellen (siehe auch Abschnitt 'DOS-Tips' Seite 41)

DOS!

?????? 3118 C2E
Eine 0 in dieser Zelle sorgt dafür, daß von zwei Dateien mit demselben Namen nur die erste gelöscht wird, wenn Sie mit Delete (D-Option im DOS-Menü) lediglich eine davon auslöschen wollen.

?????? 3818 EEA
Siehe 3822

?????? 3822 EEE
Die beiden letztgenannten Speicherplätze sind Bestandteil der DOS-Routine, die jeden eingegebenen Dateinamen daraufhin prüft, ob er nur aus Großbuchstaben und Zahlen besteht. Ändern Sie Ihr DOS wie folgt:
POKE 3118,33:POKE3122,123
Sie können dann auch Dateinamen verwenden, die mit Zahlen oder Kleinbuchstaben beginnen. Alle Zeichen außer Grphik und Inverse-Video sind jetzt zugelassen. (Jedoch Vorsicht mit Punkt, Doppelpunkt, Komma etc.!)

?????? 4226 1082
Gehört zum DOS. In diesem Byte steht normalerweise $105. Schreibt man hier einen anderen Wert hinein (z.B. POKE 4226,10), läßt sich damit das Disketten-Inhaltsverzeichnis verlegen, sodaß es mit der DOS-Option 'A' nicht mehr gesichtet werden kann! Näheres siehe Abschnitt 'Software-Schutz' auf Seite 46.

HPOSPO 53248 D000
Horizontale Position von PLAYERO

nur zw. Ø und 227
bei anderer Pl.Größe bis zu 4Ø und 19Ø reduziert

MOPF 53248 D000
MISSILEO/Spielfeld-Kollision

HPOSP1 53249 D001
Horizontale Position von PLAYER1

M1PF 53249 D001
MISSILE1/Spielfeld-Kollision

HPOSP2 53250 D002
Horizontale Position von PLAYER2

M2PF 53250 D002
MISSILE2/Spielfeld-Kollision

HPOSP3 53251 D003
Horizontale Position von PLAYER3

M3PF 53251 D003
MISSILE3/Spielfeld-Kollision

HPOSMO 53252 D004
Horizontale Position von MISSILEO

POPF 53252 D004
PLAYERO/Spielfeld-Kollision

HPOSM1 53253 D005
Horizontale Position von MISSILE1

P1PF 53253 D005
PLAYER1/Spielfeld-Kollision

HPOSM2 53254 D006
Horizontale Position von MISSILE2

P2PF 53254 D006
PLAYER2/Spielfeld-Kollision

HPOSM3 53255 D007
Horizontale Position von MISSILE3

P3PF 53255 D007
PLAYER3/Spielfeld-Kollision

MOPL 53256 D008
MISSILEO/PLAYER-Kollision

SIZEPO 53256 D008 Ø ,2 = normale Größe
Größe von PLAYERO 1 = doppelte Breite
3 = vierfache Breite

M1PL 53257 D009
MISSILE1/PLAYER-Kollision

SIZEP1 53257 D009 s.o.
Größe von PLAYER1

M2PL 53258 D00A
MISSILE2/PLAYER-Kollision

SIZEP2 53258 D00A s.o.
Größe von PLAYER2

M3PL 53259 D00B
MISSILE3/PLAYER-Kollision

SIZEP3 53259 D00B s.o.
Größe von PLAYER3

POPL 53260 D00C
PLAYERO/PLAYER-Kollision

SIZEM 53260 D00C
Größe aller MISSILE-Objekte

P1PL 53261 D00D
PLAYER1/PLAYER-Kollision

GRAFPO 53261 D00D
Graphik von PLAYERO

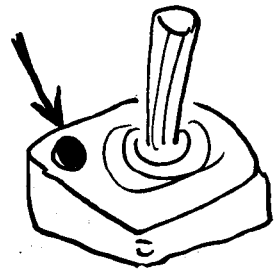
GRAFP1 53262 D00E
Graphik von PLAYER1

P2PL 53262 D00E
PLAYER2/PLAYER-Kollision

GRAFP2 53263 D00F
Grafik von PLAYER2

P3PL 53263 D00F
PLAYER3/PLAYER-Kollision

GRAFP3 53264 D010
Graphik von PLAYER3



TRIGO 53264 D010
Steuerknüppel, Trigger0 (Feuer-Knopf)

GRAFM 53265 D011
Graphik aller MISSILE-Objekte

TRIG1 53265 D011
Steuerknüppel, Trigger1

COLPMO 53266 D012
Farbe von PLAYER-MISSILE0

TRIG2 53266 D012
Steuerknüppel, Trigger2

COLPM1 53267 D013
Farbe von PLAYER-MISSILE1

TRIG3 53267 D013
Steuerknüppel, Trigger3

PAL 53268 D014
Indikator, ob PAL- oder NTSC-Ausführung des Betriebssystems

??? (eigene Untersuchungen zeigten, daß sowohl beim PAL- als beim NTSC-OS eine 1 in diesem Byte steht)

Besserer Test:

?PEEK(58383) = 249
?PEEK(65528) = 214 → PAL-Version

(Evtl. andere Werte beim neuen OS 'ROM B' !)

COLPM2 53268 D014
Farbe von PLAYER-MISSILE2

COLPM3 53269 D015
Farbe von PLAYER-MISSILE3

COLPF0 53270 D016
Farbe von Spielfeld 0

COLPF1 53271 D017
Farbe von Spielfeld 1

COLPF2 53272 D018
Farbe von Spielfeld 2

COLPF3 53273 D019
Farbe von Spielfeld 3

COLBK 53274 D01A
Hintergrund-Farbe

PRIOR 53275 D01B
Prioritäts-Auswahl

1 = alle Player vor Pfield
2 = Ø und 1 vor Pfield und 2 und 3
4 = Pfield vor Playern
8 = Farbreg. Ø und 1 Pfield vor Player und Farbreg. 2 und 3 Pfield
32 = Farbige Player

VDELAY 53276 D01C
Vertikales Delay (Verzögerung)

GRACTL 53277 D01D
Steuerung der Graphik
nur mit DMACTL = D400

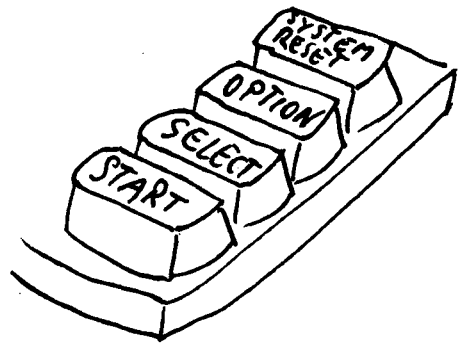
1 = nur Missiles
2 = nur Player
3 = beides

HITCLR 53278 D01E
Jede Zahl ungleich 0 bewirkt das Löschen des Kollisionsregisters.

CONSOL 53279 D01F
Abfrage, welche der Zusatztasten gedrückt wurden. 8 löscht das Register. Eine Zahl von 0-7 läßt den Konsolen-Lautsprecher klicken. X bedeutet, daß die entsprechende Taste gedrückt wurde.

Zu CONSOL:

Taste	1	2	3	4	5	6	7	8	9	0
START		X		X		X				X
SELECT	X			X	X					X
OPTION	X	X	X							X



Das AUDF -Register entspricht dem Schalter für die Soundkanäle 0-3. Die AUDCn sind die Steuer-Register für die Lautstärke.

AUDF1 53760 D200
Sound-Kanal 1: Frequenz

AUDC1 53761 D201
Sound-Kanal 1: Lautstärke

AUDF2 53762 D202
Sound-Kanal 2: Frequenz

AUDC2 53763 D203
Sound-Kanal 2: Lautstärke

AUDF3 53764 D204
Sound-Kanal 3: Frequenz

AUDC3 53765 D205
Sound-Kanal 3: Lautstärke

AUDF4 53766 D206
Sound-Kanal 4: Frequenz

AUDC4 53767 D207
Sound-Kanal 4: Lautstärke

AUDCTL 53768 D208
Audio-Steuerung

Anmerkung: +L+ = Register nur lesbar
+S+ = Register schreiben
+SL = Register schreiben/lesen

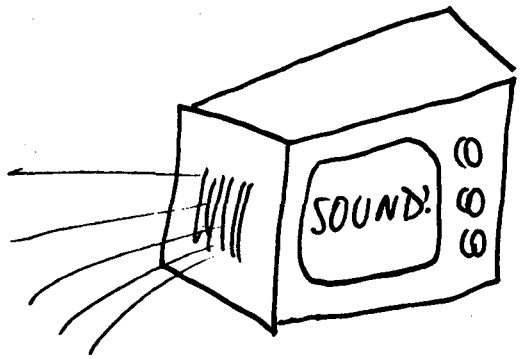
+L+ KBCODE 53769 D209
Tasten-Code (764)

+S+ STIMER 53769 D209
Starte Timer (Rücksetzen der NF-Frequenzteiler auf die AUDF-Werte)

+L+ RANDOM 53770 D20A
Zufallszahlen-Generator

+S+ SKRESET 53770 D20A
Port Status (53775) zurück.
Rücksetzen des Status der seriellen Schnittstelle in SKCTL (53775).

Port0 53760 D200 Poti 0
⋮
Port7 53767 D207 Poti 7
ALLPOT 53768 D208 Read 8 line pot. port state



+S+ POTGO 53771 D20B
Starte die POT-SCAN-Sequenz.
(Siehe Abschnitt: 'Schnittstellenprogrammierung Teil 2' auf Seite24)
+S+ SEROUT 53773 D20D
Serielle Schnittstelle, Ausgabe.

+L+ SERIN 53773 D20D
Serielle Schnittstelle, Eingabe.

+L+ IRQST 53773 D20E
IRQ-Interrupt Status

+S+ IRQEN 53774 D20E
Ermöglichen von Interrupt Request. Wird hier ausgeschaltet, arbeiten alle Tasten nicht mehr, die sonst einen Interrupt auslösen können (BREAK, SYSTEM RESET,...). 0 schaltet aus, 16 ein.

+L+ SKCTL 53775 D20F
(562) Steuerung und Status der seriellen Schnittstelle.

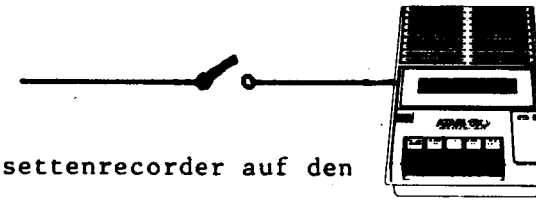
+L+ SKSTAT 53775 D20F
Lesen des Status der seriellen Schnittstelle.

+++ Für 53776-54015 wie 53760-53775 +++

+SL PORTA 54016 D300
Liest oder schreibt Daten von/an Steuerknüppel-Buchsen 1 und 2 wenn das Bit 2 von PACTL =1. Wenn das Bit 2 =0 werden Daten geschrieben. (632 für Buchse 1, 633 für Buchse 2, siehe weiter vorn)

+SL PORTB 54017 D301
Liest oder schreibt Daten von/an Steuerknüppel-Buchsen 3 und 4. (Wie oben). (634 für Buchse 3, 635 für Buchse 4)

+S+ PACTL 54018 D302
PORT-A-Steuerung:
60 Schaltet Kassettenmotor aus
52 Schaltet Kassettenmotor an
Wird benutzt, um Musik oder Sprache vom Cassettenrecorder auf den Lautsprecher des Fernsehers zu übertragen.



+S+ PBCTL 54019 D303
PORT-B-Steuerung:

+++ 54020-54271 wie 54016-54019 +++

+S+ DMACTL 54272 D400
(559) Direkter Speicherzugriff (DMA). Schaltet DMA ein, setzt ein- bzw. zweizeilige Auflösung für PLAYER-Objekte, schaltet P/M-Graphik ein.
Aus den folgenden Möglichkeiten müssen die entsprechenden Werte addiert und dann in 559 geschrieben werden:
nur mit GRACTL = D01D

Weites Spielfeld	3	Nur eins auswählen !
Standard-Spielfeld	2	
Schmales Spielfeld	1	
Kein Spielfeld	0	
Ermögliche Missile-DMA	4	
Ermögliche Player-DMA	8	
2 Zeilen breite PLAYER	0	
1 Zeilen breite PLAYER	16	
Einschaltung von Instruction-Fetch DMA	32	

CHACTL 54273 D401
(755) Zeichenkontrolle:
4 = Zeichen in Spiegelschrift
2 = Wie mit Atari-Taste
1 = Leere Zeichen

POKE 522,4

DLISTL/H 54274/5 D402/3
(560/1) Zeiger für die Display-Liste. Diese Adresse gibt an, wo das Betriebssystem die Anweisungen abgelegt hat, die den Modus und die Art der Daten festlegen, die auf den Bildschirm geschrieben werden.

HSCROL 54276 D404
Horizontales Verschieben von Bildern (Scrolling) wird ermöglicht.

VSCROL 54277 D405
Vertikales Scrolling wird ermöglichen.

PMBASE 54279/80 D407/8
Player/Missile-Basisadresse.

CHBASE 54281 D409
(756) Zeichensatz-Basisadresse. Man kann damit seinen EIGENEN ZEICHENSATZ programmieren ! Z.B. deutsche Umlaute!

WSYNC 54282 D40A
Warten auf horizontale Syncronisation.

VCOUNT 54283 D40B
Vertikaler Zeilenzähler.

PENH 54284 D40C
(564) Horizontle Position des Lichtgriffels.

PENV 54285 D40D
Vertikale Position des Lichtgriffels.

NMIEN 54286 D40E
Freigabe des nichtmaskierbaren Interrupts (NMI). Die Eingabe von 192 läßt den Display-Listen-Interrupt zu.

+S+ NMIRES 54287 D40F
Setze NMIST zurück.

+L+ NMIST 54287 D40F
NMI-Status.

+++ 54288-54277 wie 54272-54287 +++

AFP 55296 D800
Umwandlung ASCII → Floating Point.

FASC 55526 D8E6
Umwandlung Floating Point → Ascii.

IFP 55722 D9AA
Umwandlung Integer (ganzzahlig) → Floating Point (= Gleitkomma-Arithmetik).

FPI 55760 D9D2
Umwandlung Floating Point → Integer.

ZFRO 55876 DA44
Lösche FRO

ZFI 55878 DA46
Lösche Floating-Point-Wert.

FSUB 55904 DA60
Floating Point Subtraktion.

FADD 55910 DA66
Floating Point Addition.

FMUL 56027 DADB
Floating Point Multiplikation.

FDIV 56104 DB28
Floating Point Division.

PLYEVL 56640 DD40
Floating Point Polynom Entwicklung.

FLDOR 56713 DD89
Lade Floating Point-Wert.

FLDOP 56718 DD8D
Lade Floating-Point-Wert.

FLD1R 56728 DD98
Lade Floating-Point-Wert.

FLD1P 56732 DD9C
Lade Floating-Point-Wert.

FSTOR 56743 DDA7
Speichere Floating-Point-Wert.

FSTOP 56747 DDAB
Speichere Floating-Point-Wert.

FMOVE 56758 DDB6
Verschiebe Floating-Point-Wert.

EXP 56768 DDC0
Bilde den Floating-Point-Exponenten zur Basis e.

e^n

EXP10 56780 DDCC
Bilde den Floating-Point-Exponenten zur Basis 10.

10^n

LOG 57037 DECD
Bilde den Floating-Point-Logarithmus zur Basis e.

LOG10 57041 DED1
Bilde den Floating-Point-Logarithmus zur Basis 10.

++++++++++

Basis-Adressen für HANDLER-VEKTOREN für die residenten Handler-Routinen.

SCREEN-EDITOR (E) → E400
DISPLAY-HANDLER (S) → E410
KEYBOARD-HANDLER (K) → E420
PRINTER-HANDLER (P) → E430
CASSETTEN-HANDLER (C) → E440

09

++++++++++

BETRIEBSSYSTEM-EINSPRUNGADRESSEN

DISKIV 58448 E450
Disk-Handler-Initialisierung.

DISKINV 58451 E453
Disk-Handler-Einsprung.

CIOV 58454 E456
CIO-Utility-Einsprung (wird oft gebraucht!)

SIOV 58457 E459
SIO-Utility-Einsprung.

SETVBV 58460 E45C
Routine, zum gefahrlosen Setzen der Timer.

SYSVBV 58463 E45F
VBI-Einsprung Stufe1 (immediate, kritisch)

XITVBV 58466 E462
Einsprung zum Verlassen von Vertical Blank.

SIOINV 58469 E465
Einsprung zur SIO-Utility.

SENDEV 58472 E468
Send-enable-Routine.

INTINV 58475 E46B
Interrupt-Handler-Initialisierung.

CIOINV 58478 E4E6
CIO-Utility-Initialisierung.

BLKBDV 58481 E471
Blackboard-Modus-Einsprung.

WARMSV 58484 E474
Warmstart-Einsprung (RESET).

COLDSV 58487 E477
Kaltstart-Einsprung (Einschalten).

BEISPIEL FÜR DIE PROGRAMMIERUNG DER DREHREGLERANSCHLÜSSE:

HELLIGKEITSSTEUERUNG

Wie im Abschnitt 'SCHNITTSTELLEN-PROGRAMMIERUNG TEIL 2' schon erläutert, verfügt der ATARI-Computer im Gegensatz zu vielen Mitstreitern bereits von Haus aus über acht eingebaute A/D-Wandler. Hier lassen sich natürlich nicht nur die Drehregler (Paddles) anschließen, sondern auch alle anderen Bauelemente und Schaltungen, deren Wirkungsprinzip auf der Änderung des elektrischen Widerstandes beruht. Der Widerstandsbereich sollte etwa dem der Drehregler-Potis entsprechen und wird gegebenenfalls mit Hilfe von Serien- oder Parallel-Widerständen in den gewünschten Wirkungsbereich gerückt.

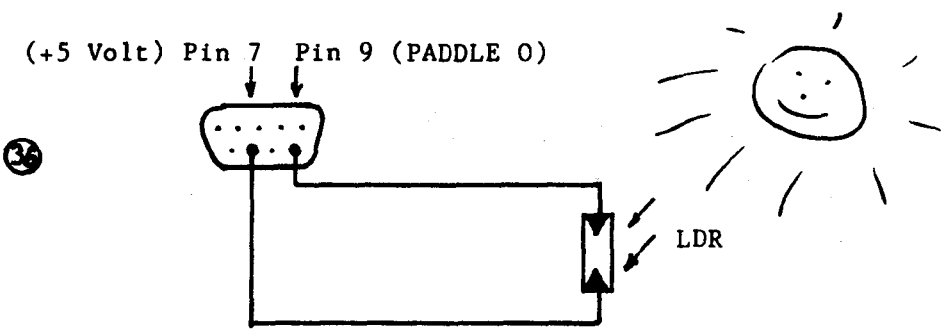
Einige Anwendungsbeispiele wären z.B.:

Widerstandsart	Anwendungsbeispiel
NTC (wärmeabh. Widerstand)	Temperaturmessung, z.B. mit Statistik und/oder Grenzwertmeldung
Potentiometer	Winkelmessung
LDR (Photowiderstand)	Helligkeitsmessung/steuerung Lichtorgel
Meß-Sonden + Flüssigkeit	Füllstandsüberwachung

Da die Unterprogramme zur Abfrage der acht Analog-Eingänge und deren Digitalisierung bereits Bestandteil des Betriebssystems sind, gestaltet sich die Programmierung derartiger Anwendungen äußerst einfach. Als Beispiel wollen wir uns einmal die HELLIGKEITSSTEUERUNG vornehmen.

In Büro- oder Laborräumen mit wechselnder Helligkeit (Sonneneinstrahlung etc.) wäre es wünschenswert, wenn sich die Bildschirmhelligkeit unseres Computersystems automatisch an das Umgebungslicht anpassen würde. Dies läßt sich durch den Anschluß eines Photowiderstands an einen der Drehregler-Eingänge erreichen (siehe Abb. 36). Abhängig von der Belichtung ändert der Widerstand seinen Wert, der wiederum zur programmgesteuerten Einstellung der Bildschirmhelligkeit verwendet wird. Jede der 16 Farben läßt sich ja bekanntlich in acht Helligkeitsstufen darstellen, was auch in diesem Fall völlig ausreicht. Solange Sie keinen LDR zur Hand haben, können Sie zur Simulation natürlich auch einen Drehregler benutzen.

Abbildung 36



Sehen Sie sich dazu das nachfolgende kleine BASIC-Programm an. Wollen wir die Helligkeit der Buchstaben in der Grundstufe (GRAPHICS 0) ändern, müssen wir den Wert vom Farbregister COLOR 1 variieren. Dies geschieht in Zeile 40, und zwar mit dem für Drehregler-Eingang PADDLE(0) gefundenen Wert, der mittels Division durch 16 in den mit SETCOLOR nutzbaren Bereich von 0 bis 14 gebracht wird.

```
10 REM  HELLIGKEITSSTEUERUNG
20 REM  MIT LDR AN PADDLE 0
30 SETCOLOR 2,2,0:SETCOLOR 4,2,0
40 SETCOLOR 1,1,INT(PADDLE(0)/16)
50 GOTO 40
```

Am sinnvollsten bringen Sie Zeile 40 in einem Unterprogramm unter, das die Bildschirmhelligkeit ausreichend oft aktualisiert.

Die Programmierung der Widerstandsabfrage und der daraus abgeleiteten Helligkeits-(oder Farb-)Änderung kann natürlich auch in Maschinensprache erfolgen. Dies ist in den folgenden zwei Anwendungsfällen sinnvoll:

1. Die Verarbeitung muß sehr schnell geschehen (z.B. für Lichtorgel-Effekte)
2. Die Routine soll interruptgesteuert im Hintergrund laufen.

Ein gleichwertiges Maschinenprogramm, das von BASIC aufgerufen werden kann, sieht folgendermaßen aus:

```
                0100 ;HELLIGKEITSSTEUERUNG IN MASCHINENSPRACHE
0000            0110        * =   $06A0      ; =dezimal 1696
06A0 68         0120         PLA
06A1 AD7002     0130         LDA  $0270      ; PADDLE0 LESEN
06A4 6A         0140         ROR  A          ; DURCH 16 TEILEN
06A5 6A         0150         ROR  A
06A6 6A         0160         ROR  A
06A7 6A         0170         ROR  A
06A8 8DC502     0180         STA  $02C5      ; IN COLOR1 ABSPEICHERN
06AB 60         0190         RTS             ; ZURUECK INS HAUPTPROGRAMM
06AC            0200         .END
```

Auch hier wird PADDLE(0) abgefragt, durch 16 geteilt und als Helligkeitswert in COLOR1 eingesetzt. Der Maschinen-Code läßt sich natürlich aus BASIC heraus generieren, wenn man ihn in eine DATA-Zeile schreibt:

```
10 FOR I=1696 TO 1696+11
20 READ A
30 POKE I,A
40 NEXT I
50 DATA 104,173,112,2,106,106,106,106,141,197,2,96
60 X=USR(1696)
70 GOTO 60
```

Auch in diesem Fall läßt sich Zeile 60 nach einmaligem Aufbau des Maschinenprogramms (READ/DATA) als Unterprogramm verwenden.

ÜBRIGENS: Belichtet man den Photowiderstand mit der Glühbirne einer (einkanaligen) Lichtorgel, lassen sich Musikstücke programmgesteuert durch einzigartige Licht- und Farbenspiele untermalen. Wer daran interessiert ist, sollte ausgiebig mit der widerstandsabhängigen Änderung der diversen Helligkeits- und Farbregister experimentieren.

Wenn die Helligkeitssteuerung unabhängig von bestimmten Programmen oder Programmiersprachen arbeiten soll, läßt man sie am Besten interruptgesteuert im Hintergrund laufen (siehe Abbildung 37).

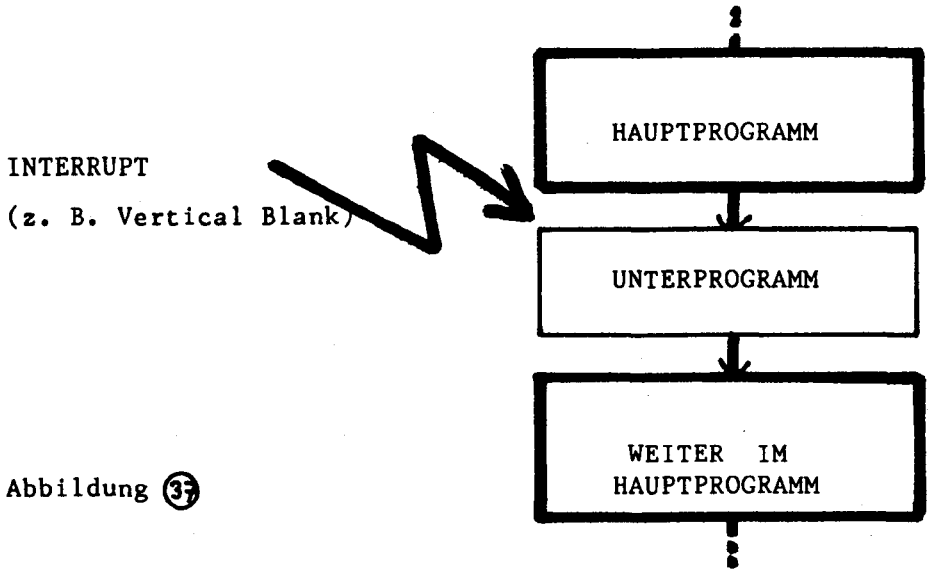


Abbildung 37

Im Beispiel dazu benutzen wir den Timer 2. Wie auch beim Timer 1 handelt es sich um einen Zwei-Byte-Zähler, der sich auf einen bestimmten Wert setzen läßt, von dem aus er heruntergezählt wird. Das Dekrementieren geschieht jeweils im Vertical Blank Interrupt, also im 50-Hz-Rythmus. Den beiden Timern (es gibt auch noch drei weitere, jedoch mit anderen Funktionen) ist jeweils eine Vektor-Adresse zugeordnet. Hier kann man die Adressen von Unterprogrammen hineinschreiben, die immer dann angesprungen werden, wenn der jeweilige Timer den Wert Null erreicht. Die Häufigkeit der Unterprogrammausführung richtet sich also nach dem Wert, von dem aus man den Timer herunterzählen läßt. In unserem Beispiel auf der folgenden Seite passiert folgendes:

1. Zeile 170-200: Das Unterprogramm wird einmalig initialisiert, indem die Vektor-Bytes von Timer 2 ($228,229) auf die UP-Startadresse ($06B0) gesetzt werden.

2. Zeile 210-220: Timer 1 wird als Ausgangswert auf 1 gesetzt.

3. Timer 2 wird im VBI herabgezählt. Bereits beim ersten Mal erreicht er den Wert Null, sodaß über seinen Vektor das eigentliche LDR-Unterprogramm angesprungen wird.

4. Zeile 250-300: Auf bereits bekannte Weise erfolgt nun Messung des Widerstands, Teilung durch 16 und Setzen von COLOR 1.

5. Zeile 310-330: Timer 2 wird wieder auf 1 gesetzt. Anschließend geht es im Hauptprogramm weiter. Bei Auftreten des nächsten Vertical Blank Interrupt wird wieder unser Unterprogramm ab Zeile 250 ($06B0) abgearbeitet.

```
                0100 ;HELLIGKEITSSTEUERUNG IM VERTICAL BLANK INTERRUPT
0000            0110        * =    $6A0
021A            0120 CDTMV2 =      $21A
0228            0130 CDTMA2 =      $228
0270            0140 PADDLO =      $270
02C5            0150 COLOR1 =      $2C5
                0160 ; INITIALISIERUNG
06A0 A9B0       0170         LDA  #$B0
06A2 8D2802     0180         STA  CDTMA2
06A5 A906       0190         LDA  #$06
06A7 8D2902     0200         STA  CDTMA2+1
06AA A901       0210         LDA  #$01
06AC 8D1A02     0220         STA  CDTMV2
06AF 60         0230         RTS
                0240 ;UNSER LDR-UNTERPROGRAMM
06B0 AD7002     0250         LDA  PADDLO
06B3 6A         0260         ROR  A
06B4 6A         0270         ROR  A
06B5 6A         0280         ROR  A
06B6 6A         0290         ROR  A
06B7 8DC502     0300         STA  COLOR1
06BA A901       0310         LDA  =$01
06BC 8D1A02     0320         STA  CDTMV2
06BF 60         0330         RTS
06C0            0340         .END
```

Als Anfangsadresse wurde bewußt $06A0 anstelle von $0600 gewählt, um Konflikte mit Hilfsprogrammen wie Monkey Wrench (tm) etc., die ebenfalls den Anfangsbereich von Page-6 benutzen, zu vermeiden.

Das gezeigte Unterprogramm-Prinzip läßt sich natürlich auch für zahlreiche andere Anwendungsfälle, z.B. für die Klangerzeugung einsetzen. - Um z.B. bei BASIC, EDITOR/ASSEMBLER, DOS etc. stets von vornherein die Helligkeitsregelung vorzusehen, kann man das gezeigte Programm auf den verwendeten Disketten als AUTORUN.SYS-Datei speichern. Nach dem 'Booten' des DOS wird es dann automatisch geladen und installiert. Assemblieren Sie dazu das gezeigte Programm oder lassen Sie es nach Eingabe der Hex-Codes vom BASIC generieren (Siehe Hilfsprogramm auf Seite 37). Gehen Sie dann ins DOS und speichern Sie den Maschinen-Code mit 'K' wie folgt ab:

```
D1:AUTORUN.SYS,06A0,06BF,,06A0   (return)
```

Wichtig!

Anmerkung:

Zur optimalen Anpassung empfiehlt sich, den größten Teil der Oberfläche des Photowiderstandes mit lichtundurchlässigem Lack bzw. Isolierband abzudecken.
→ Ausprobieren!

ROM-MODULE SELBERMACHEN !

Die Überschrift sollte nicht falsch verstanden werden: in diesem Abschnitt soll nicht erklärt werden, wie man sich eine Billig-Kopie von STAR RAIDER (tm Atari) brennt, sondern wie man seine eigenen Programme so aufbereitet, daß sie in ROMs bzw. EPROMs lauffähig sind. Vielleicht gelingt Ihnen ja eines Tages der große Wurf mit einem selbsterdachten Spiel auf ROM-Basis!

Die Grundlagen der EPROM-Programmierung und die Erfordernis spezieller EPROM-Modulkarten wurde bereits an anderer Stelle in diesem Buch ausführlich behandelt (siehe Seite 26ff). Sehen wir uns daher gleich an, wie ein Programm in einem ROM-Modul aufgebaut ist:

	linkes rechtes ROM-Modul
ROM-INIT.-ADRESSE HI	BFFF (9FFF)
ROM-INIT.-ADRESSE LO	BFFE (9FFE)
OPTIONS-BYTE	BFFD (9FFD)
KENN-BYTE = 00	BFFC (9FFC)
ROM-STARTADRESSE HI	BFFB (9FFB)
ROM-STARTADRESSE LO	BFFA (9FFA)
ANWENDERPROGRAMM ROM 2 (4K)	B000 (9000)
	AFFF (8FFF)
ANWENDER-PROGRAMM ROM 1 (4K)	
ROM-ANFANG	A000 (8000)

Abbildung 38
Speicherplatzverteilung
im ATARI-ROM-Modul

Neben dem eigentlichen Programm benötigt das Betriebssystem noch ein paar weitere Informationen über das ROM-Modul. Es muß wissen:

1. Handelt es sich um ein Diagnose-Modul (für Testzwecke)?
2. Soll beim Einschalten die Diskette 'gebootet' werden?
3. Wo liegt die Startadresse des Programms?
4. Soll das Modul nur initialisiert, oder auch gleich gestartet werden?
5. Gibt es eine Initialisierungsadresse?

Dies alles erfährt das Betriebssystem aus den sechs letzten Bytes eines ROM-Moduls (siehe oben). Liest es nach Einschalten des Computers in $BFFC bzw. $9FFC nicht den Wert 00, leitet es daraus ab, daß sich in diesem Bereich weder RAM noch ein ROM-Modul befindet.

Das Options-Byte in BFFD bzw. 9FFD gibt dreierlei an:

Bit	7	6	4	3	2	1	0	$BFFD (9FFD)
	X	0	0	0	X	0	X	

Bit 7 = 1: Dies ist ein Diagnose-Modul. Vor jeglicher Betriebssystem-Initialisierung wird zunächst das Modul initialisiert (JMP(BFFE)).
= 0: Dies ist ein 'normales' ROM-Modul.

Bit 2 = 1: Initialisiere UND starte das Modul.
= 0: Initialisiere das Modul, aber starte es nicht.

Bit 0 = 1: Lade ('boote') die Diskette.
= 0: Lade die Diskette NICHT.

Zum Beispiel: BFFD=$05=00000101 bedeutet: Dies ist kein Diagnosemodul/ starte das Modul/lade die Diskette

Die ROM-Initialisierungsadresse gibt an, wohin das Betriebssytem per JSR immer dann springen soll, wenn die RESET-Taste gedrückt oder der Computer erneut eingeschaltet wird. Für die meisten Anwendungen reicht es völlig aus, wenn hier ein Zeiger zu einem beliebigen RTS-Befehl ($60) steht.

Wichtiger ist für uns die ROM-Startadresse. Sie wird auf Drücken von RESET bzw. beim Einschalten angesprungen, sofern Bit 2 des Options-Bytes =1 ist. Sollen auf RESET bzw. Einschalten unterschiedliche Routinen angesprungen werden, müssen Sie im Anwenderprogramm eine entsprechende Abfrage der Marke WARMST ($0008) vorsehen (siehe auch Abschnitt 'MEMORY MAP ODER WAS LIEGT WO IM ATARI').

EIN KLEINES BEISPIEL:

Wir wollen die hier gewonnenen Erkenntnisse an einem kleinen Beispiel ausprobieren. Vorraussetzung ist natürlich, daß Sie über ein EPROM-Programmiergerät, vorzugsweise mit Anschlußmöglichkeit an Ihren ATARI, verfügen.

Zum Ausprobieren wollen wir uns einmal das erste Programmbeispiel aus dem Abschnitt 'Helligkeitssteuerung' vornehmen. Zugegeben, diese Routine ist mit ihren 13 Bytes nicht sehr umfangreich. Aber wenn Sie dieses einfache Beispiel zum Laufen gebracht haben, dürfte es Ihnen nicht schwer fallen, selbst komplizierteste Programme nach demselben Prinzip aufzubereiten. Erinnern wir uns: die 'Helligkeitssteuerung' bestand darin, daß wir den Wert eines an Paddle-Eingang 0 angeschlossenen Photowiderstands messen, ihn durch 16 teilen und das Ergebnis zur Steuerung der Helligkeit verwenden, indem wir es in das Farbregister COLOR 1 schreiben. Hier noch einmal der dazugehörige Maschinen-Code, jedoch bereits in den Adreßbereich des ROM-Moduls verlegt:

```
0000          10          * =   $B000     ; OBERES 4K-ROM DES ROM-MODULS
B000 AD7002   20 START     LDA   $270      ; PADDLE0 LADEN
B003 6A       30           ROR   A        ; DURCH 16 TEILEN
B004 6A       40           ROR   A
B005 6A       50           ROR   A
B006 6A       60           ROR   A
B007 8DC502   70           STA   $2C5      ; ERGEBNIS IN COLOR1 SCHREIBEN
B00A 4C00B0   80           JMP   START
```

Da das Programm natürlich kürzer als 4 kByte ist, kommen wir mit dem oberen ROM unseres Moduls aus. Wir beginnen daher bei $B000. Unser Programm reicht somit bis $B00C. Außerdem versehen wir unser EPROM mit den sechs Bytes, in denen die Information für das Betriebssystem steht:

a) Da wir keinen RTS-Code in unserem Programm verwendet haben, schreiben wir in Zelle SBFF9 den Wert $60 hinein, um ihn später mit dem in SBFFE,BFFF stehenden Vektor anspringen zu können.

b) Die Startadresse lautet $B000. In Zelle SBFFA schreiben wir daher deren niedrigeres Byte ($00) und in SBFFB das höhere Byte ($B0).

c) Ins Kenn-Byte $BFFC schreiben wir den Wert $00.

d) Soll beim Einschalten nicht der Versuch gemacht werden, die Diskette zu laden, steht im Options-Byte in Zelle SBFFD nur in Bit 2 eine 1 (Starten des ROM-Moduls). Wir schreiben hier also den Wert $04 hinein (MIT Laden der Disk hieße es $05).

e) Da das Betriebssystem für die Initialisierungsadresse zumindest einen Zeiger zu einer Zelle erwartet, in der RTS ($60) steht, verwenden wir die Adresse SBFF9. Zu diesem Zweck haben wir diesenWert hier ja extra hineingeschrieben. In $BFFE kommt also $F9 und in $BFFF $BF.

Ihr EPROM-Programmer wird wahrscheinlich nur Eingaben im 4k-Adreßbereich von 0000 bis 0FFF zulassen. Das ist jedoch nicht weiter tragisch. Geben Sie trotzdem ALLE Bytes wie angegeben ein und programmieren Sie. Nach Installation des EPROMs im Modul und im Rechner wird es dann trotzdem im gewünschten Bereich, in unserem Falle $B000-BFFF, dekodiert. Daselbe gilt für das Editieren von Maschinen-Code in einem freien Speicherbereich, z.B. ab $6000, bevor er ins EPROM 'gebrannt' wird.

Anmerkung: Für Programme von weniger als 2 kByte Länge lassen sich auch die billigeren 2k-EPROMs vom Typ 2516/2716 verwenden. Da die oberen sechs Bytes ab $BFFA (9FFA) unbedingt nötig sind, legt man so ein EPROM in den höchsten Adreßbereich eines ROM-Moduls, also $B800-BFFF bzw. $9800-9FFF. Desweiteren sind geringfügige Änderungen der Sockelbeschaltung erforderlich (siehe Seite 31).

Wie unser erstes selbstgemachtes ROM-Modul aussieht, geht aus dem Schema auf der folgenden Seite hervor:

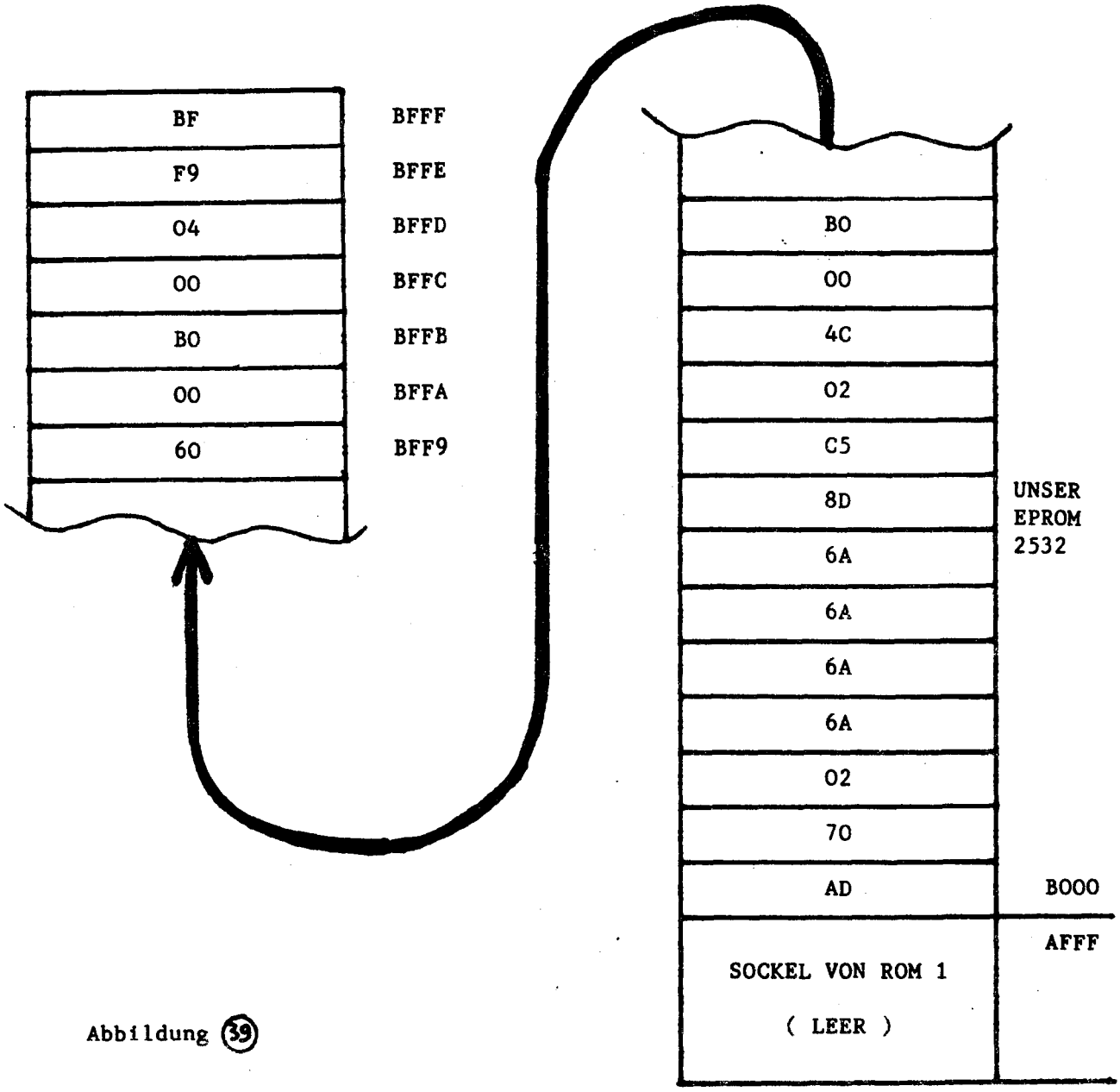


Abbildung 39

Nachdem Sie Ihr EPROM entsprechend programmiert haben, stecken Sie es bitte in den dafür vorgesehenen Sockel des EPROM-Leermoduls. Wer kein solches Modul auftreiben kann, erfährt auf Seite 32, wie sich ein ATARI-ROM-Modul zu diesem Zweck umbauen läßt. Stecken Sie das Modul in den Computer, schließen Sie an Buchse 1 Drehregler (oder LDR) an und schalten Sie das Gerät ein: auf dem blauen Bildschirm sollten Sie jetzt lediglich die Schreibmarke (Cursor) sehen. Haben Sie alles korrekt durchgeführt, sollte die Helligkeit des Cursors mit dem Widerstand erändern lassen. Falls sich nichts tut, steckt Ihr EPROM vielleicht im falschen Sockel. Falls sich nach Überprüfung noch immer nichts tut: alle Schritte nochmals genau nachvollziehen, ggf. neues EPROM programmieren.

Nachdem Sie dieses Mini-Programm als ROM-Modul zum Laufen gebracht haben, sind Sie auch für Größeres gerüstet. Eine im RAM zwar mögliche, aber auf jeden Fall 'unsaubere' Programmiertechnik sollten Sie für ROM-Software unbedingt vermeiden: Programm-Code, der sich selbst modifiziert! 'ROM' heißt schließlich READ ONLY Memory. Wenn Sie Ihre Programme stets in den richtigen Adreßbereich assemblieren und die oberen sechs Bytes mit den korrekten Werten für das Betriebssystem versehen, kann eigentlich garnichts schiefgehen. Viel Spaß!

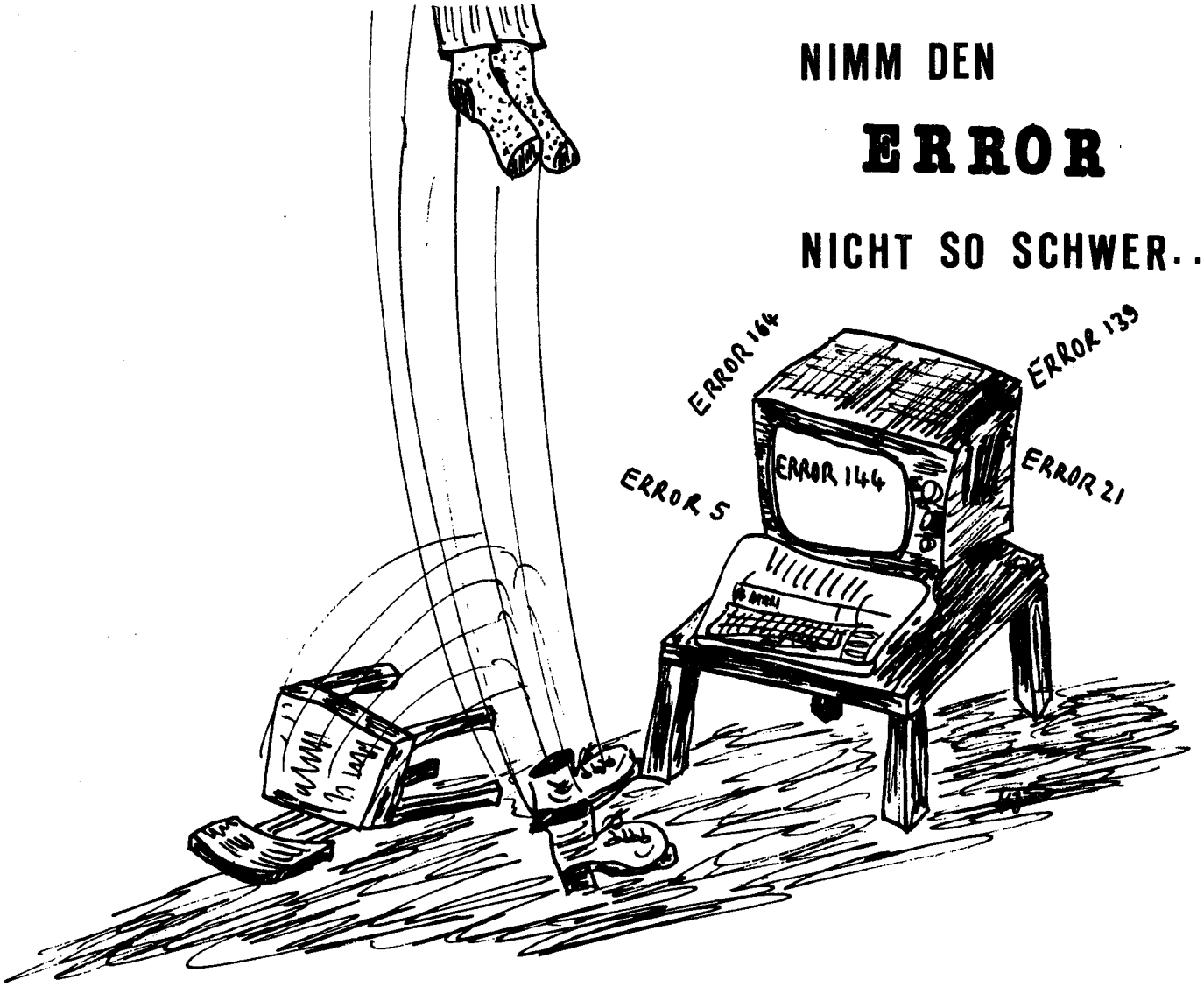
DISK TEIL 5: AUTORUN.SYS - GENERATOR

Befindet sich eine Datei mit dem Namen AUTORUN.SYS auf der Diskette in Laufwerk 1, wird sie beim Einschalten des Computers (Kaltstart) automatisch geladen und gegebenenfalls ausgeführt. Das folgende BASIC-Programm gestattet die Erzeugung eines kleinen AUTORUN.SYS-Programms, das nach dem Einschalten automatisch einen BASIC-Befehl ausführt, z.B. RUN"D:PROGNAME", ohne daß eine manuelle Eingabe erforderlich wäre. Auf diese Weise lassen sich Ihre BASIC-Programme 'idiotensicher' starten! Es sind zwar bereits verschiedene AUTORUN.SYS-Generator-Programme in Umlauf; das hier gezeigte lädt jedoch außerdem auch noch den Interface-Handler der ATARI-850-Interfacebox (sofern eine an das System angeschlossen und ZUVOR eingeschaltet ist). ACHTUNG: die AUTORUN.SYS-Datei wird sofort auf die Diskette geschrieben, die sich im Laufwerk 1 befindet ... stellen Sie daher sicher, daß Sie vorher die gewünschte Disk hineingesteckt haben! Damit das automatische Booten klappt, muß natürlich auch DOS drauf sein.

Viel Spaß beim Eintippen!

```
100 REM AUTORUN.SYS-GENERATOR
110 DIM A$(20):GRAPHICS 0
120 ? "GEBEN SIE DEN BEFEHL EIN, DER BEIM"
130 ? "BOOTEN AUTOMATISCH AUSGEFUEHRT WERDEN SOLL,";
140 ? "ZUM BEISPIEL RUN";CHR$(34);"D:PROGNAME.EXT";CHR$(34)
150 INPUT A$
160 OPEN #1,8,0,"D:AUTORUN.SYS"
170 FOR Z=1 TO 168
180 READ B
190 IF Z=136 THEN GOSUB 240
200 PUT #1,B
210 NEXT Z
220 CLOSE #1
230 END
240 FOR C=LEN(A$) TO 1 STEP -1
250 PUT #1,ASC(A$(C,C))
260 Z=Z+1
270 READ B
280 NEXT C
290 RETURN
300 DATA 255,255,106,6,255,6
310 DATA 169,80,141,0,3,169,1,141,1,3,169,63,141,2,3,169,64,141,3,3,169,5,141,6,3,141,5,3,169,0,141
320 DATA 4,3,141,9,3,141,10,3,141,11,3,169,12,141,8,3,32,89,228,16,1,96,162,11,189,0,5,157,0,3,202
330 DATA 16,247,32,89,228,48,6,32,6,5,108,12,0,96,169,193,141,33,3,169,6,141,34,3,96,231,243,51,246,207,6
340 DATA 163,246,51,246,60,246,76,228,243,172,255,6,240,9,185,234,6,206,255,6,160,1,96,140,33,3,169,228,141,34,3
350 DATA 169,155,160,1,96,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,20,226,2,227,2,106,6
360 DATA 224,2,225,2,180,6
```

++++++++++++++++++



G E W U ß T W I E :

»»» FEHLER «««

mit der FUNDGRUBE

SCHNELL GEFUNDEN + BESEITIGT !

Die meisten Fehler sind nur halb so schlimm, wenn man weiß, was man dagegen tun kann. DIE FUNDGRUBE will Ihnen dabei helfen:

TEIL 1: ALLGEMEINE TIPS ZUM UMGANG MIT FEHLERN.

TEIL 2: FEHLER-ERKLÄRUNGEN
Seine zahlreichen und präzisen Fehlermeldungen gibt Ihr ATARI leider nur als Nummern von sich. Damit Sie nicht lange in diversen 'Manuals' wühlen müssen, finden Sie hier die Fehler-Nummern im Klartext erklärt und bekommen Tips zur Fehlerbeseitigung.

TEIL 3: HILFSPROGRAMM: ATARI-BASIC-FEHLERMELDUNGEN IM KLARTEXT
Laden Sie das hier gezeigte Programm zusätzlich in den Computer: Anstelle der Fehler-Nummern gibt's dann Fehlermeldungen, die Sie verstehen!

*** DIE BEHANDLUNG VON FEHLERN IM ATARI-SYSTEM, TEIL 1: ALLGEMEINE TIPS

Im ATARI-System lassen sich die Fehlerquellen in vier Kategorien einteilen:

1. HARDWARE-FEHLER - Z.B. der Versuch, eine schreibgeschützte Diskette zu lesen, fehlerhaft arbeitende Diskettenstation etc.,
2. ÜBERTRAGUNGSFEHLER DES SERIELLEN BUS-SYSTEMS - Fehler, die bei der Datenübertragung zwischen dem Computer und einem angeschlossenen Peripheriegerät auftreten, z.B. Rahmenbit-Fehler, Zeitabschaltung (Timeout), Device NAK etc.,
3. TREIBERSOFTWARE-FEHLER - Sie werden von der Treiber-Software des angesprochenen Ein/Ausgabe-Gerätes festgestellt, wie z.B.: Datei nicht gefunden, unzulässige Laufwerk-Nummer, etc., und
4. BETRIEBSSYSTEM-FEHLER - Hierzu zählen z.B. alle möglichen Arten der Falscheingabe, wie z.B. unzulässiger Befehl, falsche Kanal-Nummer etc.

Ohne daß Sie im Teil 2 die exakte Ursache einer Fehlermeldung nachschlagen müssen, sagt Ihnen schon die Fehlernummer selbst, welchem Bereich der aufgetretene Fehler entstammt (z.T. überlappen sich diese Bereiche):

	Fehler hängt zusammen mit:
Fehlernummer von 1 bis 32:	Programmiersprache (BASIC, Assembler etc.)
Fehlernummer von 128 bis 147:	OS/Schnittstellen
Fehlernummer von 150 bis 154:	Interface ATARI 850
Fehlernummer von 160 bis 173:	Disk

Während es sich bei den Fehlermeldungen 2 bis 127 mehr um 'Warnungen' handelt, zeigen die Nummern ab 128 ($80) die 'schweren' Fehler an. Siehe auch TEIL 3 (b).

Für BASIC-Programme gilt beim Auftreten von Fehlern, daß Sie nach deren Korrektur in der Regel automatisch dort fortfahren können, wo das Programm gestoppt hat, nämlich durch Eingabe des Befehls CONT. Besser ist es natürlich, das Programm von vornherein gegen alle denkbaren Fehler abzusichern. Dazu dient der TRAP-Befehl, dessen Anwendung im ATARI BASIC Reference Manual auf Seite 23 erklärt wird.

WICHTIG:
1. Der TRAP-Befehl muß VOR der Programmstelle stehen, an der der Fehler auftreten kann.
2. Hat der TRAP-Befehl aufgrund eines Fehlers angesprochen, muß er mit einer Zahl über 32767 erneut aktiviert werden, z.B. TRAP 40000.

ÜBRIGENS: Viele Fehler, die beim Disketten- bzw. Cassetten-Betrieb auftreten, lassen sich vermeiden, wenn man für saubere Schreib/Lese-Köpfe sorgt (Cassette: regelmäßig Reinigungscassette verwenden - Diskette: siehe Seite 16ff).

Zu TEIL 2: Die Erklärungen sind für jede Fehler-Nummer nach folgendem Schema aufgebaut:

ATARI BASIC : engl./deutscher Kurzkommentar / Erläuterungen
BASIC A+ : engl./deutscher Kurzkommentar
Beispiel : (falls erforderlich)
++ Abhilfe : Ratschläge zur Fehlerbeseitigung
MICROSOFT BASIC: engl./deutscher Kurzkommentar + Erläuterungen
ACHTUNG: Das ATARI MICROSOFT BASIC verwendet viele Fehler-Nummern anders !!

ERROR 1 :

BASIC A+[2] : Break Key Abort
: Abbruch durch BREAK-Taste
Während SET 0,1 festgelegt war, hat der Benutzer die BREAK-Taste gedrückt. Dieser 'Fehler' läßt sich mit TRAP weiterverarbeiten; er verleiht dem Programmierer vollkommene Kontrolle über das System.

MICROSOFT : NEXT Without FOR
: NEXT ohne FOR
Es wurde ein NEXT-Befehl ohne den zugehörigen FOR Befehl benutzt. Dieser Fehler tritt auch auf, wenn die Variablen des NEXT Befehls bei der Schachtelung vertauscht wurden.

ERROR 2 :

ATARI BASIC: Memory Insufficient
: Speicher zu klein
Der Computer ist vom Speicherplatz her nicht mehr in der Lage, einen neuen Befehl oder eine neue Variable abzuspeichern bzw. weitere Strings zu DIMensionieren.

BASIC A+ : Mem full
: Speicher voll
Siehe ATARI BASIC.

++ Abhilfe: Programm kürzen, Programm teilen, Speicher auf Peripherie legen (z.B.Floppy), Programm optimieren (ggf. mit Komprimierungsprogramm).

MICROSOFT : Syntax
: Formfehler
Falsche Interpunktion, offene Klammern, illegale Zeichen und falsch geschriebene Befehle verursachen diesen Fehler.

ERROR 3 :

ATARI BASIC: Value Error
: Falscher Wert
Ein Wert, der ganzzahlig positiv sein sollte, ist negativ oder liegt außerhalb des vorgegebenen Bereichs.

BASIC A+ : Value Error
: Werte-Fehler
Siehe ATARI BASIC.

++ Abhilfe: Zuweisung überprüfen, ggf. INT- bzw. ABS-Funktion benutzen.

MICROSOFT : RETURN Without GOSUB
: RETURN ohne GOSUB
Ein RETURN-Befehl wurde gefunden, ohne daß bereits ein GOSUB aufgetreten ist. ist.

[2] BASIC/A+ und OS/A+ (früher: CP/A) sind eingetragene Warenzeichen von OPTIMIZED SYSTEMS SOFTWARE (OSS), 10379 Lansdale Ave., Cupertino, CA 95014 / USA. Tel.: (408) 446-3099

ERROR 4 :

ATARI BASIC: Too Many Variables
: Zu viele Variablen(namen)
Es wurde das Maximum von 128 verschiedenen Variablennamen überschritten

BASIC A+ : Too Many Vars
: Zu viele Variablen
Siehe ATARI BASIC.

++ Abhilfe: Löschen Sie alle Variablennamen, die nicht mehr in Gebrauch sind (siehe BASIC-Ref.-Manual Abschnitt 11). Verwenden Sie, wo dies ohne Überschneidung möglich ist, in verschiedenen Abschnitten Ihres Programms dieselben Variablennamen (z.B. für FOR/NEXT-Schleifen).

MICROSOFT : Out of DATA
: Zu wenig Daten
Für einen READ- oder einen INPUT -Befehl standen nicht genug DATEN zur Verfügung. Eventuell wurde ein DATA-Element oder alle Daten auf Disk bzw. Cassette vergessen.

ERROR 5 :

ATARI BASIC: String Length Error
: Stringlängen-Fehler
Versuch unter/über die DIMensionierte Stringlänge zu schreiben oder zu lesen.

BASIC A+ : String Length
: Stringlänge
Siehe ATARI BASIC.

++ Beispiel: DIMA$(2):A$(6)="A"

++ Abhilfe: Programm überprüfen, evtl. LEN-Funktion verwenden bzw. größer DIMensionieren.

MICROSOFT : Function call error
: Fehler beim Funktionsaufruf
Es wurde versucht, eine Funktion unter Benutzung illegaler Parameter auszuführen.

++ Beispiele: die Quadratwurzel einer negativen Zahl oder ein negativer LOG.

ERROR 6 :

ATARI BASIC: Out Of DATA Error
: Zu wenige DATA-Elemente
Sie haben für Ihre READ-Anweisung nicht genug Daten in der dazugehörigen DATA-Anweisung.

BASIC A+ : READ No DATA
: READ ohne DATA
Siehe ATARI BASIC.

Fortsetzg. ERROR 6

++ Abhilfe: DATA-Ausdrücke ergänzen, Anzahl der READ-Ausführungen überprüfen, evtl. wurden bei DATA-Ausdrücken Kommata vergessen. Evtl. fehlt ein RESTORE-Befehl oder er weist auf die falsche Zeile.

MICROSOFT : Overflow
: Überlauf
Eine zu große oder zu kleine Zahl ist als Ergebnis einer Funktion oder einer Eingabe aufgetreten.

ERROR 7 :

ATARI BASIC: Number Greater Than 32767
: Zahl (Zeilennummer) größer als 32767
Der Wert ist nicht positiv und ganzzahlig, oder er ist größer als 32767.

BASIC A+ : LINE /VAL - 32767
: Zeilennr./Wert - 32767
Siehe ATARI BASIC.

++ Beispiele: GOTO 50000; oder GOSUB INFO in einer Anwendung, die als Rechenergebnis für die Variable INFO auch Werte größer als 32767 zuläßt.

++ Abhilfe: Überprüfen Sie alle mathematischen Ausdrücke sowie alle Befehle, in denen Bezug auf Zeilennummern genommen wird (also GOTO, GOSUB, RESTORE), insbesondere dann, wenn diese Werte erst während des Programmablaufs errechnet werden.

MICROSOFT : Out Of Memory
: Speicher voll
Der gesamte verfügbare Speicher wurde benutzt oder reserviert. Dieser Fehler kann bei der Definition großer Felder und verschachtelten Sprüngen wie GOTO GOSUB und FOR/NEXT-Schleifen auftreten.

ERROR 8 :

ATARI BASIC: Input Statement Error
: Fehler im INPUT
Versuch,einer numerischen Variablen einen nicht numerischen Wert zuzuweisen.

BASIC A+ : INPUT/READ
INPUT oder READ bekanßm nicht den erwarteten Daten-Typ. Siehe ATARI BASIC.

++ Abhilfe: Überprüfen Sie alle Variablentypen und Eingabedaten.

MICROSOFT : Undefined Line
: Nicht vorhandene Zeile
Es wurde versucht, zu einer nicht vorhandenen Zeile zu sprinen.

ERROR 9 :

ATARI BASIC: Array Or String DIM Error
: Feld- oder String-DIM-Fehler
Das DIM-Format ist größer als 32767 (Strings) bzw. größer als 5640 (Num. Arrays), oder ein Wert liegt außerhalb des DIMensionierten Bereichs (vielleicht ist auch überhaupt keine DIMensionierung erfolgt!).

BASIC A+ : DIM
Siehe ATARI BASIC.

++ Abhilfe: Variablentabelle erstellen. Dazu gibt es diverse Cross-Reference- (XREF-)Hilfsprogramme.

MICROSOFT : Subscript Out Of Range
: Index außer Bereich
Die Benennung eines Matrix-Elements erfolgt über den DIMensionierten Bereich hinaus.

ERROR 10 :

ATARI BASIC: Argument Stack Overflow
: Argumenten-StapelSpeicher-Überlauf
Zu viele GOSUBs oder zu komplizierte Ausdrücke.

BASIC A+ : Expr Too Complex
: Ausdruck zu komplex
Siehe ATARI BASIC.

++ Abhilfe: Überprüfen auf Endlosschleifen; ggf. POP-Befehl verwenden. Ausdrücke vereinfachen, Programm kürzen.

MICROSOFT : Redefinition Error
: Mehrfach-Definitions-Fehler
Versuch eine Matrix zu DIMensionieren, obwohl sie bereits dimensioniert oder mit Default-Werten festgelegt wurde.

ERROR 11 :

ATARI BASIC: Floating Point Overflow/Underflow Error
: Gleitkomma-Überlauf
Versuch, durch Null zu dividieren oder eine Zahl größer als 1E+98 oder kleiner 1E-99 zu erzeugen.

BASIC A+ : Overflow
Siehe ATARI BASIC.

++ Abhilfe: Variablen-Zuweisungen überprüfen; evtl. fehlt Zuweisung, sodaß eine Variable noch leer ist, d.h. den Wert Null enthält. Arithmetische Ausdrücke (Formeln) überprüfen.

MICROSOFT : Division By Zero
: Division durch Null
Eine Null im Nenner eines Bruchs ist unzulässig.

ERROR 12 :

ATARI BASIC: Line Not Found
: Zeile nicht gefunden
Ein GOTO, GOSUB oder THEN zeigt auf eine nicht vorhandene Zeilennummer.

BASIC A+ : No such line
: Zeile nicht vorhanden. Siehe ATARI BASIC

++ Beispiel: Sprung zu einer REM-Zeile (schlechter Stil!), die später herausgenommen wurde.

++ Abhilfe: Sprungziel ändern, evtl. REMark Zeile einfügen.

MICROSOFT : Illegal direct
: Illegale Direkteingabe
INPUT, GET und DEF dürfen nur im Programm und nicht im Direkteingabemodus verwendet werden.

ERROR 13 :

ATARI BASIC: No Matching FOR Statement
: Kein zugehöriges FOR
Ein NEXT wurde gefunden, ohne daß zuvor ein entsprechender FOR-Befehl abgearbeitet wurde,oder FOR/NEXT-Schleifen sind nicht richtig strukturiert. (Fehlermeldung erfolgt beim NEXT, nicht beim fehlerhaften FOR!)

BASIC A+ : NEXT, no FOR
: NEXT, kein FOR
Siehe ATARI BASIC. Zusätzlich hier der Hinweis: Auch ein falscher POP-Befehl kann diese Fehlermeldung verursachen.

++ Abhilfe: Überprüfen Sie, ob die FOR-NEXT Schleifen richtig Strukturiert sind. Evtl. Sprung in eine Schleife?

MICROSOFT : Type mismatch
: Nichtübereinstimmender (Variablen-)Typ
Es ist unzulässig, Text-(String-) Variablen numerische Variablen zuzuweisen und umgekehrt.

ERROR 14 :

ATARI BASIC: Line Too Long Error
: Zeile zu lang
Die eingegebene Zeile ist länger als der BASIC-Puffer; evtl. zu komplexe Anweisungen.

BASIC A+ : Line Too Long
: Zeile zu lang
Siehe ATARI BASIC.

++ Beispiel: oft wird die Zeile zu lang, wenn man sie mit abgekürzten Befehlen (SE., G. etc.) 'vollstopft', die dann vom Interpreter in ihre volle L;nge umgewandelt werden.

++ Abhilfe: Zeile aufbrechen und auf mehrere Zeilen verteilen.

ERROR 15 :

ATARI BASIC: GOSUB Or FOR Line Deleted
: GOSUB- oder FOR-Zeile wurde gelöscht.
Ein NEXT- oder RETURN-Befehl wurde aufgefunden, dessen entsprechendes FOR

oder GOSUB seit dem letzten RUN gelöscht wurde.

BASIC A+ : Line deleted
: Zeile gelöscht
Siehe ATARI BASIC. Zusätzlich hier der Hinweis: Dieser Fehler tritt auch auf, wenn während des Programmablaufs nach GOSUB bzw. FOR ein STOP ausgefuhrt und das Programm mit CONT fortgesetzt wurde. Mit RETURN bzw. NEXT gibt es dann Schwierigkeiten.

++ Abhilfe: Fehlende Zeile wieder einfügen.

MICROSOFT : Quantity overflow
: Format-Überschreitung
Text-(String-)Variablen dürfen unter MICROSOFT BASIC 255 Zeichen nicht übeschreiten.

ERROR 16 :

ATARI BASIC: RETURN Error
: RETURN-Fehler
Ein RETURN ohne vorhergehendes GOSUB wurde gefunden.

BASIC A+ : RETURN, no GOSUB
: RETURN, kein GOSUB
Siehe ATARI BASIC. Zusätzlich hier der Hinweis: Auch falsches Benutzen von POP kann zu dieser Fehlermeldung führen.

++ Beispiel: 10 PRINT"DEMO":RETURN (ohne vorheriges GOSUB)

++ Abhilfe: Entweder RETURN entfernen oder GOSUB einfügen; evtl. wurde GOSUB übersprungen.

MICROSOFT : Formula too complex
: Ausdruck zu komplex
Eine mathematische oder String-Funktion ist zu komlex (Ausdruck in kleinere Schritte zerlegen)

ERROR 17 :

ATARI BASIC: Syntax Error
: Syntax-Fehler
Der Computer hat für ihn unverständliche Eingaben bekommen (fehlerhafte Syntax) auch Hardware-Probleme und falsche Pokes können diesen Fehler verursachen.

BASIC A+ : Bad line
: Fehlerhafte Zeile
Siehe ATARI BASIC.

++ Abhilfe: Syntax korrigieren bzw. POKEs überprüfen (Memory Map S. 53ff zu Hilfe nehmen. Hilft dies nicht, sollten Sie ein RAM-Testprogramm benutzen, um Hardwarefehler auszuscheiden.

MICROSOFT : Can't continue
: Kann nicht fortfahren
Ein CONT-Befehl im Direkt- oder Eingabemodus kann nicht ausgeführt werden, da das Programm bereits mit END beendet wurde.

ERROR 18 :

ATARI BASIC: Invalid String Character
: Falsches Textzeichen
Der Text beginnt mit einem unzulässigen Zeichen, oder der Wert in einem VAL Befehl ist nicht numerisch.

BASIC A+ : Not numeric
: Kein numerischer Wert
Siehe ATARI BASIC.

++ Abhilfe: String überprüfen.

MICROSOFT : Undefined user funktion
: Undefinierte Anwender-Funktion
Die USR-Funktion kann nicht ausgeführt werden. Der Anwender-Code enthält einen logischen Fehler, oder die USR-Startadresse wurde falsch gewählt.

ERROR 19 :

ATARI BASIC: LOAD Program Too Long
: LOAD-Programm zu lang
Es steht zu wenig RAM-Speicherplatz zur Verfügung, um das Programm zu laden.

BASIC A+ : LOAD, too big
: LADEN, zu groß
Siehe ATARI BASIC. Zusätzlich hier der Hinweis: Dieser Fehler kann auch auftreten, wenn der Zeiger LOMEM verlegt wurde.

++ Abhilfe: Speicher ausbauen (max. 48kByte) oder das Programm in Funktionseinheiten aufteilen, die immer erst dann geladen werden, wenn sie gebraucht werden.

MICROSOFT : No RESUME
: Kein RESUME
Das Ende eines Programms wurde im Error-Trapping-Modus (ON-ERROR-GOTO) erreicht.

ERROR 20 :

ATARI BASIC: Device Number Error
: E/A-Geräte-Nummern-Fehler
Die E/A-Geräte-Nummer wurde größer als 7 oder gleich 0 angegeben.

BASIC A+ : File
: Datei-Nummer
Siehe ATARI BASIC.

MICROSOFT : RESUME without error
: RESUME ohne Fehlermeldung
Ein RESUME trat vor einem ON ERROR GOTO auf.

ERROR 21 :

ATARI BASIC: LOAD File Error
: Datei-Lade-Fehler
Die Datei wurde nicht im LOAD-Format abgespeichert, oder das LOAD-Format kann unter dieser BASIC-Version nicht geladen werden (ATARI BASIC und BASIC A+ haben eine unterschiedliche Token-Schreibweise!).Auch der Versuch, mit LOAD oder ENTER Maschinenprogramme (Binary Files) zu laden fhrt zu dieser Fehlermeldung.

BASIC A+ : Not SAVE file
: Keine SAVE-Datei
Siehe ATARI BASIC.

++ Abhilfe: Die Datei mit Hilfe des ENTER- bzw. (C)LOAD-Befehls laden. Bei Benutzen von BASIC A+ UND ATARI BASIC: Dateien mit LIST abspeichern und dann mit ENTER laden (und gegebenenfalls mit SAVE wieder abspeichern. Bei Maschinenprogrammen: aus DOS mit'Binary Load' (L) laden und ggf. mit 'Run At Adress' (M) starten (bzw. mit USR aus BASIC).

MICROSOFT : FOR without NEXT
: FOR ohne NEXT
Ein NEXT Befehl wurde ohne vorhergehenden FOR-Befehl angetroffen.

ERROR 22 :

ATARI BASIC: ---

BASIC A+ : 'USING' Format
: 'USING' Formatierung
Dieser Fehler tritt auf, wenn die Länge des gesamten formatierten Strings in einem PRINT-USING-Befehl größer als 255 ist. Er tritt auch auf, wenn die Länge des einzelnen Unter-Formats für eine bestimmte Variable größer oder gleich 60 ist.

ERROR 23 :

ATARI BASIC: ---

BASIC A+ : 'USING' Too Big
: 'USING' zu groß
Der Wert einer im PRINT USING verwendeten Variablen ist größer oder gleich 1E+50.

ERROR 24 :

ATARI BASIC: ---

BASIC A+ : 'USING' Type
: 'USING'-Typ
In einem PRINT-USING-Befehl zeigt das Format an, daß eine Variable numerisch sein soll, während es sich tatsächlich um eine Text-(String-)Variable handelt (oder umgekehrt).
++ Beispiel: PRINT USING"%%%,A

ERROR 25 :

ATARI BASIC: ---

BASIC A+ : DIM Mismatch
: Nichtübereinstimmung der DIMensionierung
Der String, der mit RGET von einem Peripheriegerät geladen wurde hat eine andere Länge als die Variable, der er zugewiesen werden soll.

ERROR 26 :

ATARI BASIC: ---

BASIC A+ : Type Mismatch
: Nichtübereinstimmende (Variablen-)Typen)
Die Daten, die mit RGET von Diskette geladen werden sollen sind numerisch, die Variable, der sie zugewiesen werden aber eine String-Variable (oder umgekehrt).

ERROR 27 :

ATARI BASIC: ---

BASIC A+ : INPUT Abort
: INPUT abgebrochen
Während der Ausführung eines INPUT-Befehls hat der Benutzer CTRL-C (RETURN) eingegeben.

ERROR 28 :

ATARI BASIC: ---

BASIC A+ : Nesting
: Verschachtelung
Das Ende einer Kontrollstruktur wie ENDIF oder ENDWHILE wurde erreicht, obwohl an der Spitze des Run-Time-Stack nicht die entsprechenden Anfangsdaten liegen.

ERROR 29 :

ATARI BASIC: ---

BASIC A+ : PLAYER/MISSILE Number
: PLAYER/MISSILE-Numerierung
Die Player müssen von 0-3 numeriert sein, die Missiles von 4-7.

ERROR 30 :

ATARI BASIC: ---

BASIC A+ : PM Graphics Not Active
: P/M-Grafik nicht aktiv

Fortsetg. ERROR 30

Der Benutzer hat versucht, einen von PMGRAPHICS verschiedenen PM-Befehl zu abzuarbeiten, bevor PMGRAPHICS 1 oder PMGRAPHICS 2 ausgeführt wurde.

ERROR 31 :

ATARI BASIC: ---

BASIC A+ : Fatal System Error
: Fataler Systemfehler
Keine weitere Beschreibung. OSS bittet um schriftliche Benachrichtigung unter Angabe der näheren Umstände, falls diese Fehlermeldung einmal auftreten sollte.

ERROR 32 :

ATARI BASIC: ---

BASIC A+ : End Of 'ENTER'
: Ende von 'ENTER'
Diese Fehlermeldung resultiert aus einem Programmabschnitt wie:
SET 9,1: TRAP line : ENTER dateiname
wenn der ENTER-Befehl ohne Störungen beendet wird.

++++++++++++++++++++

Die folgenden Fehler treten in Zusammenhang mit den benutzten E/A-Geräten wie Drucker, Cassetten oder Diskette (auch Tastatur und Bildschirm zählen dazu!) auf. Sie sind mit 'OS: ' gekennzeichnet. Fehler, die speziell beim Disketten-Betrieb entstehen können, sind mit 'Disk:' gekennzeichnet.

ERROR 128 :

OS : BREAK key abort
: Abbruch durch BREAK
Während einer E/A-Operation wurde BREAK gedrückt und die Ausführung so unterbrochen.
Das Drücken von BREAK kann unter DOS2.OS manchmal auch zum System-'Absturz' führen.

ERROR 129 :

OS : IOCB already open
: IOCB schon offen
Der IOCB ist bereits für andere Zwecke geöffnet, oder ein OPEN befindet sich innerhalb einer Schleife...

++ Abhilfe: Ggf. Schließen und Neuöffnen der Datei. Evtl. wurde CLOSE vergessen, oder die IOCB-Nummer ist bereits vergeben (IOCB = E/A-Steuerblock, siehe Mem.-Map Seite 55f).

ERROR 130 :

OS : Non exsistent device
: Nicht vorhandenes E/A-Gerät
Es wurde versucht, ein E/A-Gerät anzusprechen, das nicht in der Handler-Tabelle steht, z.B. "R:", ohne daß zuvor der RS232-AUTORUN.SYS-Handler für die Interface-Box ATARI 850 geladen wurde. Auch ein gänzliches Vergessen des E/A-Gerätenamens (PROGNAME statt D1:PROGNAME) führt zu dieser Fehlermeldung.

++ Abhilfe: Prüfen Sie: Ist die Interface-Box eingeschaltet? Ist die angesprochene Hardware (korrekt) angeschlossen? Bei Benutzung der RS232-Schnittstelle: ist o.a. AUTORUN.SYS geladen? Ist der benötigte Handler vielleicht nur geladen, jedoch nicht initialisiert?

ERROR 131 :

OS : IOCB write only
: IOCB nur zum Schreiben
Es wurde versucht, von einer zum Schreiben geöffneten Datei zu lesen, oder das angesprochene E/A-Gerät kann nur schreiben (z.B. Drucker).

++ Abhilfe: Datei schließen und zum Lesen bzw. zum Update (Lesen/Schreiben) öffnen

ERROR 132 :

OS : Illegal handler command
: Unzulässiger Handler-Befehl
Dies ist eine fehlermeldung der CIO, also der zentralen E/A-Routine des Betriebssystems. Der Befehl, der in den Handler gegeben wurde, ist unzulässig. Evtl. handelt es sich um einen Spezialbefehl an einen Handler, für den dieser Befehlstyp nicht vorgesehen ist.

++ Abhilfe: Überprüfen Sie Ihren XIO- oder IOCB-Befehl. und korrigieren Sie gegebenenfalls.

ERROR 133 :

OS : Device/File not open
: E/A-Gerät/Datei nicht offen
Die angegebene Datei wurde noch nicht mit Hilfe eines OPEN-Befehls geöffnet.

++ Abhilfe: Öffnen der Datei bzw. Überprüfen der Datei-Nummer.

ERROR 134 :

OS : Invalid IOCB number
: Falsche IOCB-Nummer
Sie benutzen einen unzulässigen IOCB-Index: Bei BASIC muß die Zahl zwischen 1 und 7 liegen (0 darf nicht verwendet werden); der Assembler / Editor (ATARI-ROM-Modul) benötigt ein Vielfaches von 16, jedoch kleiner als 128.

ERROR 135 :

OS : IOCB read only
: IOCB nur zum Lesen
Es wurde versucht, auf eine zum Lesen geöffneten Datei zu schreiben, oder das angesprochene E/A-Gerät kann nur gelesen werden (z.B. Tastatur).

++ Abhilfe: Datei schließen und zum Schreiben bzw. zum Update (Lesen/Schreiben) öffnen

ERROR 136 :

OS : End of file
: Ende der Datei
Das Ende der Datei ist erreicht worden. Es sind keine weiteren Daten zu lesen!

++ Umgehen des Abbruchs: TRAP-Befehl oder EOF-Funktion benutzen.

ERROR 137 :

OS : Truncated record
: Datei abgeschnitten
Dieser Fehler tritt typischerweise auf, wenn der zu lesende Datensatz größer ist als das im CIO-Aufruf angegebene Datenformat (BASIC: 119 Bytes).

++ Abhilfe: Evtl. versuchen Sie, eine Byte-orientierte (d.h. mit PUT aufgezeichnete) mit einem Datensatzorientierten Input-Befehl zu laden. Verwenden Sie GET! (Die umgekehrte Situation ist natürlich auch möglich).

Die Datensätze einer Datei werden mit $9B, dem EOL-(Zeilenende-)Zeichen abgeschlossen. Sofern Sie eine Datei mit PUT abgespeichert haben, verfügt sie über keinerlei EOL-Zeichen, solange Sie nicht selbst für die Eingabe von $9B-Codes gesorgt haben. Es kann also passieren, daß DOS versucht, einen Datensatz von der gesamten Größe der Datei einzulesen, was dann die obige Fehlermeldung auslöst.

Oder Sie versuchten, ein mit SAVE gespeichertes BASIC-Programm mit ENTER zu laden. Verwenden Sie LOAD!

ERROR 138 :

OS : Device timeout
: E/A-Geräte-Zeitabschaltung
Ein Befehl wurde vom Computer über den seriellen Bus an ein E/A-Gerät geschickt, der nicht innerhalb der vorgesehenen Zeitspanne beantwortet wurde.

Dieser Fehler kann auf verschiedene Weise zustande kommen:

(1) Es wurde die falsche E/A-Geräte-Nummer angegeben. Es wird z.B. ein E/A-Gerät angesprochen, das nicht angeschlossen, nicht angeschaltet oder defekt ist; z.B. OPEN #3,4,0,"D3:PROGNAME", obwohl Sie nur über die Disk-Laufwerke D1 und D2 verfügen.

Fortsetzg. ERROR 138

++ Abhilfe: Überprüfen Sie die Kabelverbindungen, die Kodierung der Disketten-Stationen (siehe Kodierschalter auf der Rückseite der ATARI 810). Überprüfen Sie auch sämtliche Steckverbindungen der seriellen Schnittstelle. Durch schräges Einstecken der Stecker kommt es bisweilen vor, daß in der E/A-Geräte-Buchse einzelne Anschlußstifte umknicken oder sogar abbrechen! Sollte der Fehler weiterhin auftreten handelt es sich evtl. um einen Hardware-Fehler.

(2) Die Diskette ist angeschlossen, kann den Befehl jedoch nicht rechtzeitig ausführen. Ein Problem, daß eigentlich nur in Zusammenhang mit dem (inzwischen veralteten) DOS 1 auftrat.

++ Abhilfe: Auf jeden Fall DOS 2 verwenden! Alternativ dazu gibt es inzwischen auch von dritter Seite Disketten-Betriebssysteme, wie z.B. OS/A+ (tm Optimized Systems Software) oder K-DOS (tm K-Byte).

CASSETTEN: Diese Fehlermeldung tritt beim Cassetten-Laden wegen eines Fehlers im Betriebssystem bisweilen fälschlich auf. Der Bug wurdeallerdings im neuen OS ('B'-ROMs, ab Anfang '82) beseitigt.

++ Abhilfe: Cassette etwas vorspulen, Laden nochmals versuchen.

ERROR 139 :

OS : Device NAK
: E/A-Geräte-NAK

Diese Fehlermeldung ist eine Zusammenfassung diverser nicht weiter bestimmbarer Fehler, meist resultierend aus der Angabe falscher Parameter, z.B. dem Versuch, den Disk-Sektor Null zu lesen. Auch die Verstümmelung von Daten auf ihrem Weg vom oder zum Computer kann zu dieser Fehlermeldung führen, beispielsweise bei Verwendung minderwertiger Cassetten 'NAK' steht übrigens für Negative AcKnowledgement, d.h. ein negatives, also nicht erfolgtes Quittungssignal).

++ Abhilfe: Befehls-Parameter überprüfen. Wenn's nicht hilft: Verbindungsleitungen überprüfen. Da es sich hier um einen E/A-Geräte-spezifischen Fehler handelt, sollte man auch einen Blick in das Handbuch des angesprochenen E/A-Geräts (ATARI 850 etc.) werfen.

ERROR 140 :

OS : Serial frame error
: Serieller Rahmenbit-Fehler

Das Bit 7 des POKEY-Chips (SKSTAT, siehe Memory Map) ist gesetzt, d.h. die Kommunikation zwischen Computer und Pripherie ist gestört. POKEY hat also bemerkt, daß zuviele oder zu wenige Bits pro Byte eintreffen.
Meist Hardwareproblem.

++ Sofern Hardware-Probleme ausscheiden: insbesondere bei Benutzung der seriellen System-Schnittstelle und der RS232-Schnittstelle ist darauf zu achten, daß die miteinander kommunizierenden E/A-Geräte mit gleichen Parametern arbeiten (Paritätsprüfung ja/nein, Anzahl der Start- und Stop-Bits). Dies gilt insbesondere für den Datenaustausch zwischen ATARI-Computern über Modems, wenn unterschiedliche Treiber-Software benutzt wird (für den Modem-Betrieb in unseren Landen beachte man den Hinweis auf SEITE 44 unten (Postbestimmungen).

ERROR 141 :

OS : Cursor Out Of Range
: Cursor außer Bereich
Der Cursor käme außerhalb des Grafikfensters zu liegen.

++ Abhilfe: Cursor-Position für die verwendete Graphik-Stufe überprüfen. Sofern sich die Position aus einem Rechenergebnis ergibt, sollten Sie ggf. eine Begrenzung des Wertebereichs vornehmen. Den jeweils zulässigen Wertebereich finden Sie im BASIC-Ref.-Manual auf der Innenseite des hinteren Deckblatts.

ERROR 142 :

OS : Serial overrun
: Serieller Overrun
Das Bit 5 des POKEY-Chips (SKSTAT, siehe Memory Map) ist gesetzt, d.h. der Computer ist nicht in der Lage die über den seriellen Bus eintreffenden Daten schnell genug zu verarbeiten (sehr seltener Fehler).

++ Abhilfe: Tritt er doch einmal häufiger auf, sollten Sie Ihren Computer in einer Service-Werkstatt überprüfen lassen).

ERROR 143 :

OS : Checksum error
: Prüfsummen-Fehler
Die Prüfsumme für die serielle Übertragung von Daten ist stimmt nicht! Die erhaltenen Daten sind also fehlerhaft, denn die Summe der Rahmen-Bits vom Ursprung stimmt nicht mit der vom Computer errechneten Summe überein.

++ Abhilfe: Viel läßt sich hier nicht tun. Es kann sich sowohl um die verschiedensten Software- als auch Hardware-Probleme handeln; eventuell hat lediglich eine Netzstörung 'dazwischengefunkt'. Meistens helfen ein paar erneute Versuche.

ERROR 144 :

OS : Device done error
: E/A-Geräte-Ausführungsfehler
Der Alptraum des ATARI-Besitzers! Das E/A-Gerät ist nicht in der Lage, einen zulässigen Befehl auszuführen.
Dieser gräßliche Fehler tritt bei den verschiedensten gelegenheiten auf:

(1) Es wurde versucht, auf eine schreibgeschützte Diskette zu schreiben.

++ Abhilfe: Schreibschutz-Aufkleber entfernen. Prüfen Sie aber zunächst, ob es sich nicht um eine Master-Diskette handelt, und Sie der ERROR 144 vor eventuellem Überschreiben wichtiger Information bewahrt! Oder steckt die Diskette vielleicht falsch herum im Laufwerk? (Das Etikett muß vorne/oben sein). Ist die Diskette überhaupt formatiert?

Bei 'kopiergeschützten' oder durch Kontakt mit magnetischen Gegenständen beschädigten Disketten kann das Lesen einzelner Sektoren unmöglich sein.
++ Abhilfe in diesem Fall: Hoffentlich haben Sie eine Back-Up-Diskette !
(2) Allgemein: Problem mit dem angesprochenen E/A-Gerät.

ERROR 145 :

OS : Illegal Screen Mode / Read After Write Compare Error
: Unzulässiger Graphik-Modus / Fehler beim Lesetest nach dem Schreiben
Es wurde versucht, den Bildschirm-Editor mit einer falschen Nummer für den Graphik-Modus öffnen.

++ Abhilfe: Überprüfen Sie den betreffenden GRAPHICS-Befehl bzw. das AUX2-Byte im IOCB.

ERROR 146 :

OS : Function not implemented
: Funktion nicht vorgesehen
Der HANDLER kann diese Funktion nicht ausführen, da er sie nicht kennt, z.B. PUT auf die Tastatur.

++ Abhilfe: Überprüfen Sie, ob Sie den richtigen Befehl und/oder das richtige E/A-Gerät angegeben haben.

ERROR 147 :

OS : Insufficient RAM
: Kein ausreichender RAM-Platz vorhanden

Für die gewählte Grafikstufe steht nicht genug RAM-Kapazität zur Verfügung.

++ Abhilfe: Speicher aufrüsten (ist bis zu 48k möglich) oder kleinere Grafikstufe wählen. Eine nützliche Auflistung, wieviel RAM welcher Graphik-Modus 'verschlingt', finden Sie im BASIC-Ref.-Manual auf der Innenseite des hinteren Deckblatts.

ERROR 150 - 154 :

Bei diesen 5 Fehlermeldungen handelt es sich um spezielle Fehler, die beim Betrieb der Interface-Box ATARI 850 auftreten können (in Zusammenhang mit ATARI 850 können natürlich auch ERROR 128 bis 139 auftreten).

++ Abhilfe: Einzelheiten dazu finden Sie im 850-Operator's Manual auf Seite 37/38.

ERROR 160:

Disk : Drive number error
: Laufwerknummern-Fehler
Es wurde eine illegale Laufwerk-Nummer (nicht 1 bis 8) angegeben, oder es ist kein Buffer für diese Disk angelegt worden.

++ Abhilfe: Überprüfen Sie den Datei-Aufruf und/oder Byte 1802($710), das die Laufwerknummern enthält (siehe Seite 65).

ERROR 161 :

Disk : Too many open files
: Zu viele offene Dateien
Für weitere Dateien sind keine freien Sektor-Puffer mehr verfügbar (maximal 8 offene Dateien gleichzeitig!).

++ Abhilfe: Überprüfen Sie Byte 1801($709) auf die Anzahl der benutzten Puffer.

ERROR 162 :

Disk : Disk full
: Diskette voll
Die Diskette ist voll. Es können keine Daten mehr auf die Diskette geschrieben werden.

++ Abhilfe: Ganz einfach: Eine neue Diskette einlegen! Sie haben gerade keine leere Diskette mehr im Haus? Dann helfen Ihnen vielleicht die Tips auf Seite 12 und Seite 42!

ERROR 163 :

Disk : Fatal system I/O error
: Fataler E/A-Fehler
Fehler im DOS-File-Manager.

Abhilfe: Eventuell ist das DOS auf der von Ihnen verwendeten Diskette aus irgendeinem Grund fehlerhaft. Verwenden Sie ein anderes DOS. Sollte dies den Fehler beheben, sollten Sie auch gleich auf die fehlerhafte Diskette ein 'frisches' DOS schreiben!

ERROR 164 :

Disk : File number mismatch
: Dateinummern vertauscht , fehlerhafte Sektor-Verkettung
Ein Zeiger aus einer Datei zeigt auf einen Sektor, der zu einer anderen Datei gehört. Auch dieser Fehler kann durch Kontakt der Diskette mit magnetischen E/A-Geräten (Scheren, Schraubenzieher etc.) verursacht werden. Besonders gern tritt er auch bei Verwendung minderwertiger oder abgenutzter Disketten auf.

++ Abhilfe: Während der Ausführung eines Programms sind Sie gegen diesen besonders unangenehmen Fehler praktisch machtlos. Versuchen Sie mit Hilfe der Befehle B und E des Hilfsprogramms FIX (Disk Fixer, ATARI-APX) die Sektor-Folge der beschädigten Datei wiederherzustellen. Anschließend sollten Sie sofort eine Sicherheitskopie erstellen!

ERROR 165 :

Disk : Filename error
: Dateinamen-Fehler
Der Dateiname enthält illegale Zeichen. Zulässig sind lediglich alphanumerische Zeichen, Stern und Fragezeichen.
++ Abhilfe: Auf korrekte Schreibweise überprüfen!

ERROR 166 :

Disk : Point Data Length
: Fehler beim POINT-Befehl
Der Bytezähler im POINT-Befehl war größer als 125 (einfache Aufzeichnungsdichte) oder 253 (doppelte Aufzeichnungsdichte, nicht bei ATARI 810).

++ Abhilfe: Parameter im POINT-Befehl verbessern.

ERROR 167 :

Disk : File locked
: Datei(name) gesperrt
Versuch, in eine gesperrte Datei zu Schreiben.

++ Abhilfe: Datei freigeben (UNLOCK).

a) im DOS: option 'G'
b) in BASIC: XIO36,#1,0,0,"D1:PROGNAME"

ERROR 168 :

Disk : Device command invalid
: Ungültiger E/A-Geräte-Befehl
Sie haben der Software-Schnittstelle des angesprochenen E/A-Geräts einen unzulässigen Befehl erteilt.

++ Abhilfe: Schauen Sie in die Dokumentation des betr. E/A-Geräts und korrigieren Sie gegebenenfalls Ihre Eingabe.

ERROR 169 :

Disk : Directory full
: Inhaltsverzeichnis ist voll
Das Inhaltsverzeichnis kann keine neuen Dateinamen mehr aufnehmen (Maximum = 64 Eintragungen).

++ Abhilfe: Löschen Sie alle nicht mehr benötigten Dateien. Wo möglich, fassen Sie mehrere Programme oder Datenblöcke zu einer Datei zusammen. Damit sparen Sie Dateinamen. Sollte dieser Fehler ausgerechnet während der Bearbeitung eines BASIC-Programms auftreten (MEM.SAV geht dann ja auch nicht mehr!), kann man sich zum Löschen von Dateien auch hier eines XIO-Befehls bedienen:

XIO33,#1,0,0,"D1:PROGNAME"

ERROR 170 :

Disk : File not found
: Datei nicht gefunden
Sie haben versucht, auf eine Datei zuzugreifen, die nicht im Inhaltsverzeichnis vermerkt ist.

++ Abhilfe: Es wurde wahrscheinlich ein falscher Dateiname eingegeben. Im Zweifelsfall den DIR-Befehl vom OS/A+ bzw. den DOS-'A'-Befehl benutzen. Steckt die richtige Diskette im Laufwerk? Erweiterungszeichen (.BAS etc.) vergessen? Dateinamen zu hastig eingetippt? (Nur keine Hektik!)

ERROR 171 :

Disk : POINT Invalid
: Ungültiger POINT-Befehl
Es wurde versucht, auf ein Byte zu 'POINTen', das hinter dem Ende der Datei liegt. Oder die Datei war nicht für UPDATE geöffnet.

Abhilfe: Überprüfen Sie Ihre POINT-Parameter, bzw. im zweiten Fall den OPEN-Befehl oder das AUX1-Byte des IOCBs, den Sie zum öffnen der Datei benutzt haben

ERROR 172 :

Disk : Illegal append
: APPEND hier nicht zulässig
Sie haben eine DOS-1-Datei zum Anhängen von Daten (Append) geöffnet, arbeiten jedoch unter DOS 2. Dies ist nicht zulässig.

++ Abhilfe: Kopieren Sie die DOS-I-Datei mit Hilfe von DOS 2 auf eine DOS-II-Diskette.
Hinweis: OS/A+ ist ebenfalls zu DOS I nicht kompatibel.

ERROR 173 :

Disk : Bad sectors At Format
: Schlechte Sektoren beim Formatieren
Der Floppy-Disk-Kontroller hat während des Formatierens einer Diskette fehlerhafte Sektoren entdeckt.

++ Abhilfe: Nehmen Sie eine andere Diskette! Wenn jegliches Formatieren scheitert, kann ein Fehler am Laufwerk vorliegen. Tückischerweise treten hier bisweilen auch Wärmefehler auf, die sich erst nach einigen Betriebsstunden zeigen. Hin und wieder ist auch der Kopf-Andruckfilz des 810-Laufwerks verantwortlich (siehe Seite 16: Kopfreinigung).

BOOT ERROR (ohne Nummer):

Sie versuchen eine Diskette zu laden, auf der sich werder DOS noch eine spezielle Lade-Routine befindet.

++ Abhilfe: Bitte prüfen Sie: Ist die Diskette überhaupt formatiert?
Haben Sie versäumt, die DOS-Files drauf-zuschreiben? (Dies kann JEDERZEIT nachge-holt werden, solange noch 81 Sektoren frei sind!)
Handelt es sich um eine Daten-Diskette?
Versuchen Sie versehentlich, die Rückseite zu laden?

DEM FEHLER AUF DER SPUR, TEIL 3: HILFSPROGRAMME

(A) DER FEHLERMELDER

Es gibt bereits verschiedene Software, die anstelle von Fehlernummern Mitteilungen im im Klartext ausgibt, z.B. OS/A+, BASIC/A+ (tm OSS Inc.) und K-DOS (tm Kay Enterprises). Aber auch die ATARI-BASIC-Benutzer brauchen nun auf diesen Komfort nicht mehr zu verzichten!

Geben Sie das BASIC-Programm 'FEHLERMELDER' auf der folgenden Seite ein und speichern Sie es als LIST-Datei ab. Sie können dieses nützliche Hilfsprogramm dann bequem an Ihre Programme anhängen, indem Sie es mit ENTER laden. Da der FEHLERMELDER nur die Zeilennummern ab 32700 belegt, dürfte es kaum zu Konflikten mit Ihren eigenen Programmen kommen. Ganz an den Anfang Ihres Programms setzen Sie dann noch den Befehl: TRAP 32700 (WICHTIG!!) Ergeben sich nun während des Programmablaufs irgendwelche Fehler, bekommen Sie außer der Nummer auch noch eine entsprechende Textmitteilung. Darunter erscheint die fehlerhafte Zeile, sodaß Sie mit Hilfe der Cursor-Steuerung sofort eine Korrektur vornehmen können. Eine echte Hilfe!

(B) WEITERVERARBEITUNG DER FEHLER-CODES IN MASCHINENSRACHE.

Wie eingangs bereits erwähnt, beginnen die 'ernsten' Fehlermeldungen erst bei Nummer 128 ($80). Ihre eigene Fehlerauswertung (falls erforderlich) läßt sich daher auf einfache Weise folgendermaßen durchführen:

```
JSR CIOV    ; SPRUNG INS OS BEI $E456
TYA         ; BEENDIGUNGS-CODE HOLEN
BMI FEHLER  ; UND PRUEFEN, OB   $80 (DANN FEHLER)
```

Bei der Weiterverarbeitung der genauen Fehler-Ursache hilft die folgende Tabelle:

(Der Status-Code steht übrigens in $30)

Status-Code (hex)	Meldung
01	Erfolgreiche Operation
80	Abbruch mit BREAK-Taste
81	IOCB bereits geöffnet
82	E/A-Einheit nicht vorhanden
83	IOCB nur zum Schreiben geöffnet
84	Unzulässiger Befehl
85	E/A-Einheit oder Datei nicht offen
86	unzulässige IOCB-Nummer
87	IOCB nur zum Lesen geöffnet
88	Ende der Datei
89	Abgeschnittener Datensatz
8A	Zeitabschaltung der E/A-Einheit
8B	E/A-Einheit quittiert den Befehl nicht
8C	Serieller Bus: Rahmenbit-Fehler
8D	Cursor außer Bereich
8E	Serieller Bus: Daten kamen zu schnell
8F	Serieller Bus: Prüfsummen-Fehler
90	Peripherie-Fehler: Befehl nicht ausführbar
91	Unzulässige Nummer für die Graphik-Stufe
92	Funktion ist im Handler nicht eingebaut
93	Der RAM reicht für diese Graphik-Stufe nicht aus

```
32700 REM HIER BEGINNT DER FEHLERMELDER
32701 GRAPHICS 0+16:? :? :? "AUF FEHLER #";PEEK(195);" GESTOSSEN !!":CLR :RESTORE 32703
32702 FZ=PEEK(187)*256+PEEK(186):TRAP 32705:DIM FEHLER$(40)
32703 READ FNUM,FEHLER$:IF FNUM=PEEK(195) THEN ? :? FEHLER$:GOTO 32706
32704 GOTO 32703
32705 ? :? "NICHT DEFINIERTER FEHLER"
32706 ? :? "VERURSACHT DURCH FOLGENDE ZEILE:"
32707 LIST FZ
32708 DATA 2,NICHT GENUG RAM!
32709 DATA 3,WERT AUSSER GUELTIGEM BEREICH
32710 DATA 4,ZU VIELE VARIABLEN
32711 DATA 5,STRINGLAENGEN-FEHLER
32712 DATA 6,ZU WENIG DATEN
32713 DATA 7,ZEILENNUMMER UEBERSCHREITET 32767
32714 DATA 8,NICHTNUMERISCHER WERT!
32715 DATA 9,DIMENSIONIERUNGSFEHLER (ARRAY/STRING)
32716 DATA 10,ZU VIELE GOSUBS/AUSDRUCK ZU GROSS
32717 DATA 11,GLEITKOMMA-UEBER/UNTERLAUF
32718 DATA 12,ZEILE NICHT GEFUNDEN
32719 DATA 13,KEIN PASSENDER 'FOR'-BEFEHL
32720 DATA 14,ZEILE IST ZU LANG
32721 DATA 15,GOSUB- ODER FOR- ZEILE GELOESCHT
32722 DATA 16,RETURN OHNE GOSUB GEFUNDEN
32723 DATA 17,DER COMPUTER VERSTEHT NICHT!
32724 DATA 18,UNZULAESSIGES STRING-ZEICHEN
32725 DATA 19,LADE- PROGRAMM IST ZU LANG
32726 DATA 20,E/A-GERAETE-NR. >7 ODER =0
32727 DATA 21,FALSCHER DATEI-TYP
32728 DATA 128,MIT BREAK-TASTE ABGEBROCHEN
32729 DATA 129,IOCB IST SCHON OFFEN!
32730 DATA 130,DIESE E/A-EINHEIT EXISTIERT NICHT
32731 DATA 131,LESE-BEFEHL AN NUR-SCHREIB-EINHEIT
32732 DATA 132,FUER ANGEGEBENE EINHEIT UNZULAESSIG
32733 DATA 133,GERAET BZW. DATEI NICHT OFFEN
32734 DATA 134,UNZULAESSIGE IOCB-NUMMER
32735 DATA 135,SCHREIB-BEFEHL AN NUR-LESE-EINHEIT
32736 DATA 136,DAS DATEI-ENDE IST ERREICHT
32737 DATA 137,DATENSATZ IST ZU LANG
32738 DATA 138,E/A-GERAET ANTWORTET NICHT
32739 DATA 139,SEKTOR 0 / UEBERTRAGUNGSFEHLER
32740 DATA 140,SERIELLER BUS: RAHMENBIT-FEHLER
32741 DATA 141,CURSOR AUSSER ZULAESSIGEM BEREICH
32742 DATA 142,SERIELLER BUS: DATEN KAMEN ZU SCHNELL
32743 DATA 143,SERIELLER BUS: PRUEFSUMMEN-FEHLER
32744 DATA 144,DISKETTE IST SCHREIBGESCHUETZT!
32745 DATA 145,GRAFIK-MODUS- ODER LESETEST-FEHLER
32746 DATA 146,FUNKTION IM HANDLER NICHT VORGESEHEN
32747 DATA 147,ZU WENIG RAM FUER DIESEN GRAPHIK-MODUS
32748 DATA 160,ILLEGALE LAUFWERK-NUMMER
32749 DATA 161,ZU VIELE OFFENE DATEIEN
32750 DATA 162,DISKETTE IST VOLL!
32751 DATA 163,FATALER E/A-FEHLER (NICHT BEHEBBAR)
32752 DATA 164,FEHLERHAFTE SEKTOR-VERKETTUNG
32753 DATA 165,ILLEGALER DATEI-NAME
32754 DATA 166,FER POINT FALSCHE LAENGE ANGEGEBEN
32755 DATA 167,DIESE DATEI IST GESPERRT
32756 DATA 168,UNZULAESSIGER E/A-BEFEHL
32757 DATA 169,INHALTSVERZEICHNIS IST VOLL!
32758 DATA 170,DATEI NICHT GEFUNDEN
32759 DATA 171,POINT IST HIER NICHT ERLAUBT
32760 DATA 173, FEHLER BEIM FORMATIEREN
```

BASIC-PROGRAMME AUFPOLIEREN: EINBAUFERTIGE SOUND-ROUTINEN

Kaum ein Computer gestattet eine so vielseitige und einfache Programmierung von Klängen aller Art wie der ATARI.
Auch Ihre eigenen BASIC-Programme wirken besser und erhalten oft mehr Bedienkomfort, wenn Sie Fehlermeldungen, Grenzwerte, das Drücken von Tasten etc. mit akustischen Signalen anzeigen. Hier finden Sie einige Anregungen. Ergänzen Sie jeweils am Ende einer Routine RETURN und rufen Sie sie mit GOSUB auf.

```
10 SPHAERENKLAENGE
20 FOR A=1 TO 3:FOR I=1 TO 255 STEP 3
30 SOUND 1,2*I,10,10:SOUND 2,5*I+7,10,10
40 NEXT I:NEXT A
```

****

```
10 REM TORPEDO
20 FOR I=255 TO 1 STEP -1:SOUND 1,I,8,14:NEXT I
```

****

```
10 REM AKUSTISCHE BESTAETIGUNGEN F. RICHTIG- U. FALSCHEINGABE
20 REM BEISPIEL 1
30 FOR H=15 TO 0 STEP -1:FOR Z=30 TO 0 STEP -5:SOUND 0,Z,10,H:NEXT Z:NEXT H
40 REM BEISPIEL 2
50 FOR H=15 TO 0 STEP -2:FOR Z=50 TO 30 STEP -1:SOUND 1,Z,10,H:NEXT Z:NEXT H:SOUND 1,0,0,0
60 REM BEISPIEL 3
70 FOR BIM=14 TO 0 STEP -0.1:SOUND 1,80,10,BIM:SOUND 2,81,10,BIM:NEXT BIM
80 REM BEISPIEL 4
90 FOR A=1 TO 3:FOR I=4 TO 3 STEP -1:FOR B=14 TO 0 STEP -1:SOUND 1,I*40,10,B:SOUND 2,I*40+1,10,B
95 NEXT B:NEXT I:NEXT A
99 REM GEWUENSCHTE SOUND-ROUTINE MIT RETURN ABSCHLIESSEN U. ALS UNTERPROGRAMM AUFRUFEN
```

****

```
10 REM MUENZ-EINWURF
20 F=40
30 FOR L=8 TO 0 STEP -0.2
40 FOR V=1 TO C:SOUND 0,F,10,V:NEXT V
50 FOR V=2*L TO 1 STEP -1:SOUND 0,F,10,V:NEXT V:NEXT L
```

****

```
10 REM KLINGEL
20 FOR A=1 TO 2:FOR B=1 TO 40:SOUND 1,30,10,8:SOUND 2,0,2,8
30 FOR E=1 TO 2:SOUND 1,15,10,8:SOUND 2,0,2,8:NEXT E:SOUND 1,0,0,0:SOUND 2,0,0,0:NEXT B
40 FOR C=1 TO 100:NEXT C:NEXT A
```

****

```
10 REM KONSOLEN-SUMMER
20 FOR A=1 TO 5:FOR I=1 TO 50
30 POKE 53279,0
40 NEXT I
50 POKE 66,0:? CHR$(253):NEXT A
```

BÜCHER UND ZEITSCHRIFTEN FÜR ATARI-COMPUTER

(A) B Ü C H E R

K.+ H. Inman: The ATARI Assembler (engl.)
Reston Publishing, - ISBN 0-8359-0236-6
(Einführung in 6502-Maschinensprache und Assembler-Programmierung anhand ATARI's Editor/Assembler-Cartridge).

L. Scanlon: 6502 Software Design (engl.)
H.W. Sams & Co.,Inc., - ISBN 0-672-21656-6
(Allgemeine Einführung in die 6502-Programmierung. Enthält viele fertige Maschinenprogramm-Beispiele, die sich auch für ATARI eignen)

Ken Uston: Beating The Top 16 Video Games (engl.)
The New American Library Inc., 1633 Broadway, New York, NY 10019
(Ein Buch für Game-Freaks: Gewinnstrategien für Pac-Man (tm), Asteroids (tm), Centipede (tm) und andere Automaten-Spiele; größtenteils auf die Computer-Version dieser Spiele übertragbar).

J.D. Mulliken: Pac-Man (tm) - The Ultimate Key To Winning (engl.)
Running Press, Philadelphia, - ISBN 0-89471-173-3
(Gewinnstrategien speziell für Pac-Man)

D. Kiesenberg: Programmieren in FORTH (deutsch)
T&T, D. Kiesenberg, PF 579, 46 Dortmund 1
(Für AIM-65, jedoch auch für ATARI geeignet)

Castlewitz/Chisausky: VISICALC (R) - Home and Office Companion (engl.)
Osborne/McGraw-Hill Inc., - ISBN 0-931988-50-0
(Sammlung mit gut dokumentierten Anwendungsbeispielen für Visicalc (tm))

COMPUTE!'s First Book Of ATARI (engl.) - ISBN 0-942386-00-0
(Die wichtigsten ATARI-Beiträge aus COMPUTE!-Jahrgang '80)

COMPUTE!'s Second Book Of ATARI (engl.) - ISBN 0-942386-06-X
(Vorher unveröffentlichte Artikel, hauptsächlich Hilfsprogramme, Programmiertechniken und Spiele-Beispiele)

Bill Wilkinson: Inside ATARI DOS (engl.) - ISBN 0-942386-02-7
(Ausführl. Erklärungen u. dokumentiertes Source-Listing zum DOS2.OS

COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403

(B) ZEITSCHRIFTEN

a) Aussschließlich ATARI-Computern gewidmet:

ANTIC - The Atari (tm) Resource (engl.)
ANTIC Publishing, 297 Missouri St., San Francisco, CA 94107

A.N.A.L.O.G. 400/800 Magazine (engl.)
P.O. Box 615, Holmes, PA 19043

b) Zeitschriften mit regelmäßigen Beiträgen für ATARI-Computer:

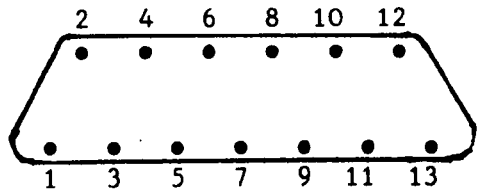
COMPUTE!, BYTE, Creative Computing, Microcomputing (ehem. Kilobaud), Interface Age, MICRO.
(Ohne Anspruch auf Vollständigkeit. Auch von ATARI selbst gibt es eine ganze Reihe guter Dokumentation, wie z.B. das Operating Systems User's Manual u.v.a.)

SCHNITTSTELLEN-PROGRAMMIERUNG TEIL 3

(A) DIE SERIELLE SCHNITTSTELLE

Da sich für eigene Entwicklungen die Benutzung der parallelen Schnittstelle wesentlich einfacher gestaltet als die der seriellen System-Schnittstelle, soll hier nur ein grober Überblick erfolgen.

Alle Peripherie-Geräte werden mit dem Computer über die asynchrone serielle System-Schnittstelle verbunden. Der Datentransfer geschieht mit 19200 Baud und einem Datenformat von 8 Bit, wobei das niederwertigste Bit zuerst übertragen wird. Dabei entspricht eine Spannung von 0 Volt log.0 und 4 Volt log.1. Es sind getrennte Ein- und Ausgangsleitungen vorhanden, und zwar DATA IN (zum Computer) und DATA OUT. Die Daten-Ausgabe über DATA OUT wird von der CLOCK-OUT-Leitung gesteuert. Immer wenn diese Leitung auf log.1 geht, nimmt DATA OUT den Pegel des nächsten zu übertragenden Bits an. Nach der halben Übertragungszeit des Bits geht CLOCK OUT wieder auf log.0 zurück. Darüberhinaus sind noch einige weitere Steuerleitungen vorhanden, wie der Blick auf die Einbaubuchse der seriellen Schnittstelle zeigt:



Anschlußbuchse der seriellen System-Schnittstelle

Anschl.	Funktion	
1	CLOCK IN	Eingangs-Takt
2	CLOCK OUT	Ausgangs-Takt
3	$\overline{\text{DATA IN}}$	Eingangs-Daten
4	GND	Masse
5	DATA OUT	Ausgangs-Daten
6	GND	Masse
7	COMMAND	*
8	MOTOR CONTROL	Motor-Steuerg.(Cassette)
9	$\overline{\text{PROCEED}}$	*
10	+5V/READY	
11	AUDIO IN	Audio ein (Cassetten-Recorder)
12	+12V	
13	$\overline{\text{INTERRUPT}}$	*

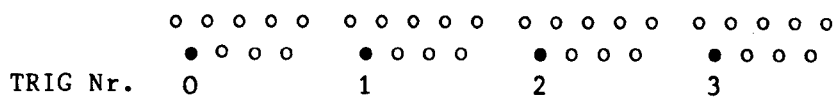
CLOCK OUT: Takt zur Steuerung der seriellen Ausgabe (siehe oben).
DATA IN: Datenleitung zum Computer (Eingang).
GND: Signal- und Abschirmungsmasse.
DATA OUT: Datenleitung vom Computer (Ausgang).
COMMAND: Ist normalerweise log.1 und geht auf log.0, wenn ein System-Befehl übertragen wird.
MOTOR CONTROL: An/Aus-Steuerung des Cassetten-Recorders.
+5V/READY: Zeigt an, daß der Computer eingeschaltet und zur Übertragung bereit ist. Zusatzschaltungen können aus dieser Leitung bis zu 50 mA ziehen.
+12V: Stellt ggf. 12 Volt für Peripheriegeräte zur Verfügung.
*$\overline{\text{CLOCK IN}}$, $\overline{\text{PROCEED}}$ und $\overline{\text{INTERRUPT}}$ werden unter der momentan verwendeten Betriebssystem-Version noch nicht benutzt.

SCHNITTSTELLEN-PROGRAMMIERUNG TEIL 3

(B) DIE TRIG-EINGÄNGE

Während die Ermittlung der Steuerknüppel-Werte vom Schnittstellen- Baustein PIA 6520 übernommen wird (siehe Seite 20ff), geschieht die Abfrage der roten 'Feuerknöpfe' von den vier TRIG-Leitungen des GTIA-Chips, der ansonsten für die graphische Aufbereitung des Bildschirms zuständig ist. Zwar lassen sich diese vier Leitungen nicht wie bei der PIA wahlweise als Ein- oder Ausgang programmieren; dafür stehen sie jedoch für eigene Anwendungen als zusätzliche Eingangskanäle mit besonderen Eigenschaften zur Verfügung.

Hier liegen die TRIG-Eingänge auf den Front-Buchsen:



Der Anschlu geschieht also jeweils an Pin 6.

Intern sind die TRIG-Eingänge wie der PIA-Port A beschaltet (s.Seite 23)

Der Zustand der einzelnen TRIG-Leitungen wird in vier verschiedenen Speicherzellen abgelegt. Da jeweils nur Bit 0 benutzt wird, ist dies zwar etwas verschwenderisch, erleichter jedoch die Programmierung, insbesondere in Maschinensprache. Es werden die Adressen D010 bis D013 benutzt:

TRIG0-3	UNBENUTZT	D0

D0 = 0: Knopf wurde gedrückt
D0 = 1: Knopf wurde nicht gedrückt

TRIG-Nr.	Register-Adr.	OS-Schattenregister
0	D010	0284
1	D011	0285
2	D012	0286
3	D013	0287

Die Abfrage geschieht einfach durch Lesen der entsprechenden Register. Steht dort der Wert 0, heißt dies: Knopf gedrückt; steht dort eine 1, wurde der Knopf - oder was immer man dort angeschlossen hat - nicht gedrückt.

Die Besonderheit dieser Eingänge liegt in der Möglichkeit, bei Bedarf eine Signal-Speicherung (sog. Latch) vorzunehmen. Normalerweise werden TRIG0 bis TRIG3 direkt vom Mikroprozessor gelesen. Ist jedoch Bit 2 des Graphik-Steuerregisters GRACTL ($D01D, dez. 53277) auf log.1 gesetzt, wird der Übergang von TRIG-Eingängen auf log.0 dauerhaft gespeichert. Und zwar so lange, bis Bit 2 von GRACTL wieder auf log.0 gesetzt wird. Dies ist besonders für Anwendungen interessant, bei denen die TRIG-Abfrage in Interrupt- oder Unterprogramm-Routinen vorgenommen wird, deren Auswertung erst später erfolgen kann.

ANMERKUNG: Da GRACTL jedoch auch noch andere Aufgaben hat, darf nicht das gesamte Byte auf $00 bzw.$FF gesetzt werden:

GRACTL	UNBENUTZT	D2	D1	D0

D2 = 1: Signalspeicherung EIN
D1 = 1: DMA-Freigabe f. PLAYER-Reg.ster
D0 = 1: DMA-Freigabe f. MISSILE-Reg.

(Die P/M-DMA-Freigabe erfordert das Setzen der Bits in GRACTL und DMACTL)

EPROMS MIT ATARI 'BRENNEN': PROGRAMMER 400/800

Da der Autor an der Entwicklung des PROGRAMMER 400/800 mitgewirkt hat und das Programmieren von EPROMs für den ATARI-Computer von allgemeinem Interesse sein dürfte, folgt hier eine kurze Beschreibung dieses Gerätes.

Der PROGRAMMER 400/800 gibt dem ATARI-Besitzer die Möglichkeit, MIT Hilfe seines Computers und FÜR seinen Computer EPROMs zu programmieren. Diese Bausteine lassen sich innerhalb des Systems vielseitig verwenden, z.B. zum Einsatz in ROM-Modulen (s.S. 32 u. 78) oder in der Disketten-Station (s.S. 7). Als erforderliche Ausrüstung zum Betrieb des PROGRAMMER 400/800 wird ein ATARI 400 bzw. 800 mit mindestens 32 kByte, Disk oder Cassettenrecorder und das ATARI-BASIC-ROM benötigt. Das Gerät wird mit drei Steckern einfach an die Frontplatten-Buchsen angeschlossen. Den Rest erledigt das mitgelieferte Betriebssystem. Nachdem es geladen ist, gibt der Anwender den verwendeten EPROM-Typ an. Bei Bedarf kann auch ein bestimmter Adreßbereich vorgegeben werden, in dem ROMs geladen und editiert werden können; andernfalls wählt die Software automatisch eine Startadresse. Anschließend meldet sich das System mit seinem Hauptmenü:

```
* B E F E H L S U E B E R S I C H T *

      RAM-STARTADRESSE(HEX)= 0000

  L LEERTEST          V VERGLEICHSTEST
  P PROGRAMMIEREN     A EPROM LESEN
  S STARTDATEN        R EPROM --> RAM
  B BYTE - MODUS      T TEILPROGRAMMIERG.
  C RAM/ROM LESEN     W RAM SCHREIBEN
  E ENDE              - NOCH FREI

BITTE WAEHLEN SIE:
```

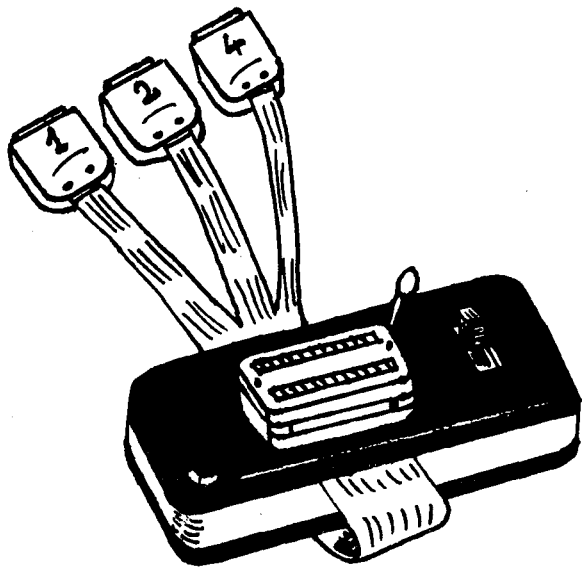
Dem Programmierer stehen somit folgende Möglichkeiten zur Verfügung:

* LEERTEST: Prüfung, ob das EPROM gelöscht ist.
* PROGRAMMIEREN von kompletten EPROMs gemäß RAM-Vorlage oder Master-ROM.
* STARTDATEN ändern: durch Verlegen des RAM-Bereichs, der später in das
* EPROM übertragen wird, lassen sich z.B. der Inhalt zweier 2k-EPROMs in ein 4k-EPROM programmieren (und umgekehrt).
* BYTE-PROGRAMMIERUNG: Programmierung von einzelnen Bytes, auch nachträglich in bereits programmierte EPROMs.
* BLOCK-PROGRAMMIERUNG: Programmierung einzelner Teilbereiche eines EPROMs.
* VERGLEICHSTEST auf Übereinstimmung von EPROM- und RAM-Inhalt (auch für Vergleich von ROMs bzw. RAMs geeignet.
* AUFLISTEN von ROMs/EPROMs oder beliebigen RAM-Bereichen (Hex/ASCII-Dump).
* EINLESEN von ROMs/EPROMs in RAM (auch mehrere Bausteine hintereinander).

Die Steuer-Software ist so gestaltet, daß der Anwender bei Bedarf auf einfache Weise eigene Erweiterungen vornehmen kann, z.B. das Abspeichern von ROM-Inhalten auf Diskette etc. Im Menü ist zu diesem Zweck ein Platz freigehalten worden.

Der PROGRAMMER 400/800 besitzt einen Umschalter zur wahlweisen Programmierung von EPROMs mit 2 kByte (Texas Instr. 2516, INTEL, HITACHI etc. 2716) und 4 kByte (Texas Instr. 2532).

Es sind auch passende EPROM-Leermodule lieferbar, die - genau wie die ATARI-ROM-Module - in den dafür vorgesehenen Schacht des Computers gesteckt werden. Wie Sie Ihre eigenen Programme aufbereiten müssen, damit sie als ROM-Modul lauffähig sind, finden Sie in diesem Buch auf Seite 78 ff.



PROGRAMMER 400/800

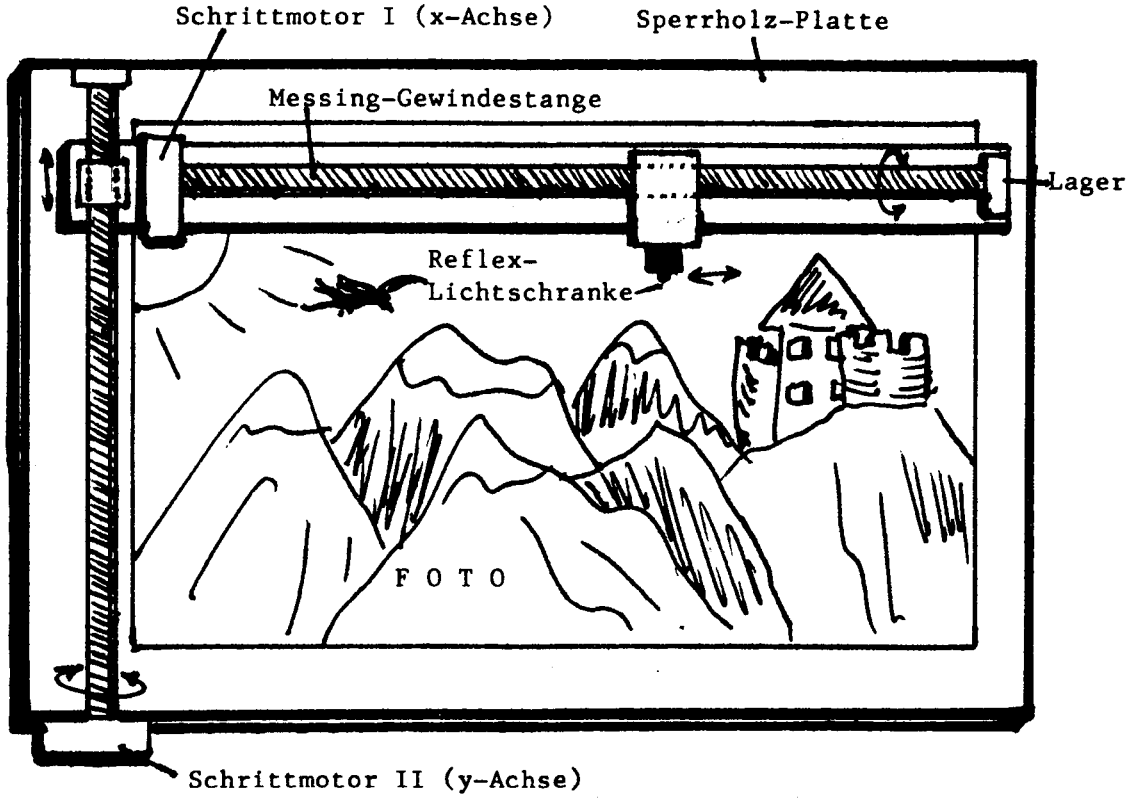
Zum Programmer 400/800 wird ein ausführliches deutsches Handbuch mitgeliefert. Bezugsquellen werden auf Anfrage gerne mitgeteilt.

ZUBEHÖR FÜR ATARI-COMPUTER: EINIGE ANREGUNGEN ZUM SELBSTBAU

Die hier vorgestellten Ideen sollen Ihnen einige Anregungen geben, was Sie als 'Erfinder' alles mit Ihrem Atari-Computer anstellen können. Es handelt sich dabei lediglich um Denkanstöße, die sich aber ohne weiteres in die Praxis umsetzen lassen.

IDEE Nr.1: FOTO-ABTASTUNG

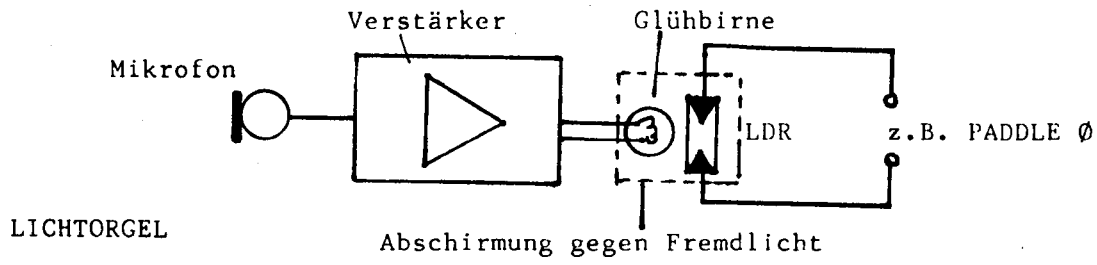
Dieses Gerät kann dazu dienen, Fotos abzutasten, zu digitalisieren und in den Computer einzuspeisen. Sie lassen sich dann auf dem Bildschirm darstellen. Der Aufbau ist einfach: Die PIA der Steuerknüppel-Schnittstelle steuert zwei Schrittmotoren, die inzwischen recht preiswert angeboten werden. Über billige Gewindestangen (z.B. Messing, M4) positionieren sie eine Reflex-Lichtschranke. Eine derartige Lichtschranke besteht aus einem Sender (LED) und einem Empfänger (Fototransistor oder Fotodiode); diese sind in geneigtem Winkel nebeneinander angeordnet, sodaß ein Schluß der Lichtschranke nur durch Reflektion an einer nahe gelegenen Fläche geschehen kann. Für unsere Anwendung nutzen wir den Teil der Kennlinie des Empfängers aus, der bei unterschiedlicher Helligkeit der Reflektionsfläche unterschiedliche Werte liefert. Schrittweise wird Zeile für Zeile abgetastet und die Graustufe des Bildpunktes, der gerade unter der Reflex-Lichtschranke liegt, über einen der Drehregler-Eingänge in den Computer eingelesen. Es sind 8 Helligkeitsstufen möglich, was eigentlich ausreichen sollte. Die Auflösung richtet sich nach der gewählten Grafik-Stufe. Durch Zählen der Schrittmotor- Impulse weiß der Computer stets, welcher Bildpunkt gerade ausgewertet wird.



IDEE Nr.2: DISKO-LICHTORGEL

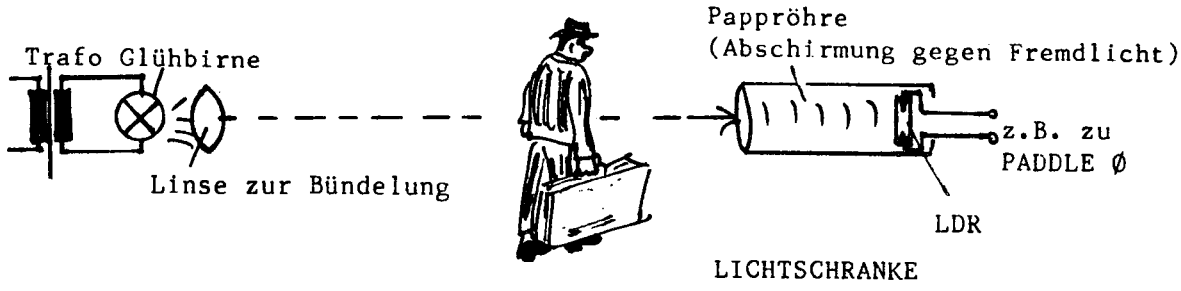
Die üblichen Mehrkanal-Lichtorgeln mit ihren flackernden bunten Glühlampen sind inzwischen ja wirklich etwas langweilig geworden. Wie wärs mit Licht- und Farbeffekten vom Farbfernsehgerät? Der hier beschriebene Zusatz zu Ihrem Atari-Computer könnte der Knüller für Ihre nächste Party werden.

Das Prinzip zeigt die folgende Abbildung. Der vom Mikrofon registrierte Schall steuert über einen kleinen Verstärker eine Taschenlampen-Glühbirne, deren Helligkeit im Rhythmus der Musik schwankt. Diese Helligkeitsänderungen wiederum bewirken Widerstandsschwankungen in einem LDR (Photowiderstand), der mit einem der Drehregler-Eingänge verbunden ist (siehe 'Helligkeitssteuerung' Seite 74 ff). Abhängig vom momentanen Widerstandswert kann man nun die Farb- und Helligkeitsregister ändern. Damit lassen sich die phantasievollsten Effekte erreichen, insbesondere in Kombination mit allen möglichen graphischen 'Kunstw erken' auf der Mattscheibe. Damit der Bildschirm schnell genug reagiert, sollte die Programmierung in Maschinensprache geschehen.



IDEE Nr.3: LICHTSCHRANKE

Die folgende Idee könnte z.B. für Geschäftsleute interessant sein, die einen ATARI im ihrem Laden haben. Sie können automatisch das Interesse eines Besuchers wecken, indem Sie, ausgelöst durch sein per Lichtschranke angekündigtes Erscheinen, mit dem Computer eine Botschaft an ihn richten. Nebenbei kann natürlich noch zu statistischen Zwecken eine Zählung vorgenommen werden. Lenken Sie durch akustische Signale den Blick des Besuchers auf den Bildschirm. Verwenden Sie entweder Computer-Sounds oder schalten Sie, gesteuert von der Lichtschranke, mit POKE 54018,52 den Cassettenrecorder ein (Endlos-Cassette verwenden). Bei nochmaligem Passieren der Lichtschranke läßt sich der Recorder mit POKE 54018,60 wieder abschalten. Besonders effektvoll kann man hier die inzwischen für ATARI von verschiedenen Abietern verfügbare Sprachsynthese einsetzen. Anstelle des LDR am Drehregler-Port und einer sichtbaren Lichtquelle läßt sich auch ein Infrarot-Phototransistor an einem der Steuerknüppel-Eingänge verwenden. Als Sender dient dann eine Infrarot-Leuchtdiode. Durch Verwendung zweier Lichtschranken läßt sich auch noch die Richtung des Passierenden ermitteln und programmtechnisch weiterverarbeiten.



	NORM	XL
Boot	F1E4	C403
Bootspez	F2F4	C5AE
Beep 61530	F05A	FDFE
Print X=LO Y=HI		C642

ZU GUTER LETZT

Unterscheidung: $FFFF : $C0 → XL

NACHLESE ZUR MEMORY MAP

Weit oben im Betriebssystem befinden sich die folgenden drei Einsprung-Adressen, die für jeden Maschinensprache-Programmierer interessant sein dürften:

XL

F24A EGETCH $F63E Zum Einlesen einer Zeile wie INPUT in BASIC.
Es muß jedoch jedes Zeichen einzeln mit JSR $F63E eingelesen werden.
Das jeweilige Zeichen steht dann auch im Akku.
Als letztes Zeichen einer Zeile muß $9B (RETURN) angegeben werden.

F2B0 EOUTCH $F6A4 Bringt ein Zeichen aus dem Akku auf den Bildschirm.
Arbeitet wie PUT#6 in BASIC,
X- und Y-Register werden verändert.

F302 KGETCH $F6E2 Bringt den ATASCII-Wert einer gedrückten Taste in den Akku.
Arbeitet wie GET# in BASIC.

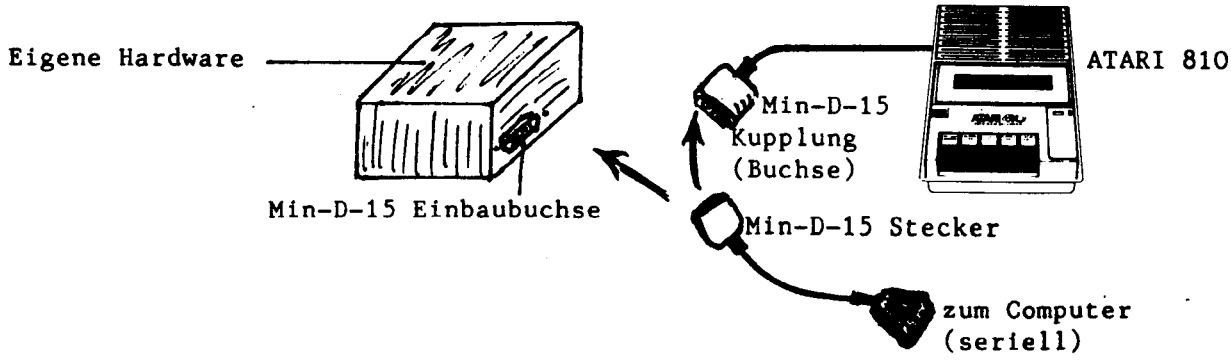
Diese Adressen decken sich mit dem ATARI-OS-Listing (Display Handler 10-30-78). Es ist durchaus möglich, daß diese Routinen in anderen OS-Versionen an anderer Stelle liegen. Die genaue Funktion muß experimentell gefunden werden.

++ Auf Seite 70 ist der Druckfehlerteufel tätig geworden und wurde zu spät entdeckt. Bei SKCTL handelt es sich um ein Register, in das geschrieben wird. Statt +L+ muß dort also +S+ stehen (bitte ausbessern).

Netzteilprobleme: In letzter Minute erreicht uns die Nachricht, daß es in Zukunft stärkere Trafos für die Disk 810 geben soll ...

Stecker für die serielle Schnittstelle: Auch der Autor ist bemüht, welche aufzutreiben. Wer Hardware zum Anschluß an den seriellen System-Bus entwickeln möchte, kann sich zwischenzeitlich mit folgendem Trick helfen:

Schneiden Sie das Kabel vom Cassettenrecorder ab. Besorgen Sie sich die gut lieferbaren 15-poligen Min-D-Steckverbinder (s.S. 43, 2b) und die passenden Gegenstücke und verbinden Sie das Kabel wieder mit dem Recorder, indem Sie die Min-D-Kupplung einfügen. Ihre Eigenentwicklungen werden ebenfalls mit diesem Steckverbinder-Typ ausgerüstet und lassen sich nunmühelos an den Computer anschließen.



Sektor-Killer: Viele Software-Anbieter schützen ihre kostspieligen Entwicklungen durch Aufbringen defekter Sektoren auf die Diskette, womit das unerlaubte Kopieren erheblich erschwert wird. Auch Sie können Ihre Programme mit diesem Trick schützen: die gewünschten Sektoren müssen lediglich bei einer Drehzahl bespielt werden, die außerhalb des zulässigen Bereichs liegt. Mehr sei hier nicht verraten: sämtliche Information dazu finden Sie in diesem Buch (Drehzahl: Seite 17. Sektor-Information: Seite 64). Geben Sie jedoch acht, daß Sie nicht Teile Ihres Programms 'killen'! Auch sei hier nochmals ausdrücklich vor Mißbrauch derartiger Methoden gewarnt.

BASIC-Tip: Wer Programme für 'Nicht-Computeristen' schreibt, muß damit rechnen, daß bei Eingaben das Drücken der RETURN-Taste vergessen wird und der Benutzer dann nicht weiter weiß. Es geht aber auch ohne RETURN. Besonders dann, wenn nur ein einziges Zeichen eingegeben werden muß. Beispielsweise die Eingabe von 'Ja' oder 'Nein' (J/N):

```
10 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
20 IF CHR$(A)="J" THEN 50
30 IF CHR$(A)="N" THEN 60
40 GOTO 10
```

Will man nur abfragen, ob irgendeine Taste gedrückt wurde, gehts noch einfacher:

```
10 IF PEEK(764)=255 THEN 10
20 REM  weiter nach Tastendruck
```

Am Besten sofort ausprobieren. Diese beiden Methoden der Eingabe-Anforderung sind sehr nützlich!

Einfache Steuerknüppel-Abfrage: Wenn es z.B. darum geht, sich durch endlos lange Ja/Nein-Menüs zu tasten, läßt sich dies sehr gut mit dem Steuerknüppel erledigen. Zum Beispiel nach rechts = ja und nach links = nein. Da die TRIG-Leitungen der Drehregler mit einigen Anschlüssen der Steuerknüppel identisch sind, läßt sich dies auf einfache Weise wie folgt programmieren (Beispiel):

```
10 A=PTRIG(0)-PTRIG(1)
20 IF A=-1 THEN PRINT "NEIN"
30 IF A=1 THEN PRINT "JA"
40 GOTO 10
```

GANZ ZUM SCHLUß NUN NOCH EINE INTERESSANTE USR-ADRESSE:

Probieren Sie einmal

```
X=USR(61530)     !!
```